精通GUI图形界面编程

施晓红　周　佳　编著

北京大学出版社

北　京

内 容 提 要

本书主要介绍MATLAB的图形绘制和用户图形界面实现技术。全书主要由三个部分组成：MATLAB语言介绍、MATLAB二维和三维图形绘制方法以及GUI的开发与应用。书中通过大量的实例深入浅出地介绍了MATLAB二维、三维曲线和曲面图形的绘制方法以及图形用户界面的设计和编程，可以帮助MATLAB设计人员完成各种特征数据的可视化并建立良好的图形界面与用户进行交互，使MATLAB强大的计算和设计功能得以充分体现。

本书适用于MATLAB的使用开发人员、大中院校师生以及广大的业余爱好者阅读，可作为相关专业的教材或参考资料。

图书在版编目(CIP)数据

精通GUI图形界面编程/施晓红，周佳编著．—北京：北京大学出版社，2003.1
ISBN 7-301-06102-1
I．精… II．①施… ②周… III．计算机辅助计算--软件包，MATLAB—程序设计
IV．TP391.75

中国版本图书馆CIP数据核字(2002)第107185号

书　　名：精通GUI图形界面编程
著作责任者：施晓红　周　佳
责 任 编 辑：黄庆生
标 准 书 号：ISBN 7-301-06102-1/TP·0702
出　版　者：北京大学出版社
地　　址：北京市海淀区中关村北京大学校内　100871
电　　话：编辑部62765013　发行部62750672　出版部62754962
网　　址：http://cbs.pku.edu.cn
电 子 信 箱：xxjs@pup.pku.edu.cn
印　刷　者：河北省滦县滦兴书刊印刷厂
发　行　者：北京大学出版社
经　销　者：新华书店
787毫米×1092毫米　16开本　16.5印张　416千字
2003年1月第1版　2003年1月第1次印刷
定　　价：24.00元

前　　言

MATLAB 是一种高效的工程计算语言，它在数值计算、数据处理、自动控制、图像处理、神经网络、小波分析、金融分析等方面有着广泛的应用。MATLAB 系统不但提供大量涉及各个工程领域的工具箱来简化科学计算、工程设计和分析等工作，而且提供具有自身特点的编程语言，可以轻松地实现大量数据的分析、处理及显示任务。

一般而言，用户总希望将数据或设计结果用图形来表示，以使数据的特征或性能能够直观地体现出来。对于一般的高级语言程序来说，绘制图形，尤其是根据计算结果所得的不规则图形是一项较为复杂的工作，用户只有在对该语言有了较为深入的了解后才能迅速准确地绘制所需的图形，而 MATLAB 面向对象的图形技术使用户可以轻松实现自身数据或处理后数据的绘制任务。

使用 MATLAB 提供的大量图形设计技术，用户无需了解图形实现的细节内容，有时甚至只需一条简单的指令就可以绘制非常复杂的图形。另外，用户还可以根据需要来规划 MATLAB 的图形外观，使得绘图结果完全符合用户的需求。

从当前的软件趋势来看，友好的图形界面已经成为应用程序的基本交互入口，为此，MATLAB 提供了对用户图形界面（GUI）的支持，使 MATLAB 开发的程序可以为更多的用户所接受。

本书主要介绍 MATLAB 图形系统的开发与应用，其内容可以分为三个大部分：MATLAB 语言介绍、MATLAB 二维和三维图形绘制方法以及 GUI 的开发与应用。读者可以通过本书十二个学时的学习掌握全书的内容，具体安排如下：

第一章（第一学时）是本书的基础，主要向读者介绍 MATLAB 系统的基础知识以及 MATLAB 语言的基本程序设计方法；

第二章（第二学时）将在第一学时的基础上使读者了解 MATLAB 语言的一些高级编程方法；

第三章（第三学时）介绍了 MATLAB 绘图的基本方法和过程；

第四章（第四学时）介绍了 MATLAB 二维基本和特殊图形的绘制方法；

第五章（第五学时）介绍 MATLAB 三维基本曲线、曲面图形和特殊图形的绘制方法，在此基础上读者可以在第六学时中学习 MATLAB 三维图形的可视化技术，包括镜头、灯光和透明度等；

第七章（第七学时）介绍 MATLAB 图形对象和句柄的概念以及使用方法；

第八章（第八学时）介绍 MATLAB 用户图形界面（GUI）的概念和基本创建过程，以及 GUI 开发工具 GUIDE 的使用方法和基本的 GUI 界面设计方法；

第九章（第九学时）介绍 GUI 的深入编程方法，包括理解文件和回调函数编程等；

第十章（第十学时）给出了四个典型的 GUI 创建和实现过程，包括完整的 M 文件代码；

第十一章（第十一学时）给出了 MATLAB 系统辨识工具箱 GUI 的具体使用方法；

第十二章（第十二学时）介绍了 MATLAB C/C++图形库的编译链接方法。

本书由国防科大王德军策划，张一鸣、王鹏主编，施晓红、周佳编著，参予编写的还有

徐飞、肖峰、沈辉、陆昌辉等。

由于作者水平有限，错误或疏漏之处难免，敬请读者不吝赐教。E-mail:daisyjia@163.net。

作　者

2002年5月

目　　录

第 1 章　MATLAB 语言入门

MATLAB 语言通常被称为是一种“演草纸式的科学计算语言”，由于它的计算是基于矢量描述的，因而在工程领域中有着广泛的应用。本章作为本书的语言基础，将主要介绍 MATLAB 的一些基础、常用的知识，包括 MATLAB 的简单介绍、MATLAB 6.0 软件的开发环境及其用户界面、MATLAB 矩阵和数组的基本运算方法等。

1.1　MATLAB 系统简介

MATLAB（matrix laboratory）本意为矩阵实验室，最初是单纯开发用于矩阵计算的，但经过这些年的迅速发展，MATLAB 已经成为一种高效的工程计算语言，在数值计算、数据处理、自动控制、图像处理、神经网络、小波分析等方面应用广泛。典型的 MATLAB 应用包括：

- 数值计算
- 计算算法
- 建模、仿真和原型
- 数据分析、检测和可视化
- 科学和工程图形
- 应用程序扩展，包括创建图形用户接口

MATLAB 采用一些常用的数学符号来表示问题及其解决方案，将计算、可视化和编程等功能集成于一个简单易用的开发环境中。MATLAB 是一种基于不限维数组数据类型的内部交互系统，既能够进行矩阵和向量计算，也能够采用特定的方法在标量语言（例如 C 和 Fortran）中编写程序。MATLAB 为用户工作平台的管理和输入输出数据提供了便利的方法，同时还提供 MATLAB 应用程序——M 文件（该文件的扩展名为 m）的扩展和管理工具。MATLAB 还采用一组被称为工具箱的特殊应用解答集，工具箱本身也是可理解的 M 文件集。工具箱的使用使得 MATLAB 能够解决许多特殊类别的问题，例如信号处理、自动控制、神经网络、模糊逻辑、小波变换、系统仿真等等。对大多数 MATLAB 用户而言，工具箱是非常重要的，它使得用户可以学习并使用一些特殊的技术。

MATLAB 系统包括五个主要部分：

- 开发环境。开发环境是帮助用户使用 MATLAB 函数和文件的工具的集合，这些工具中许多都是图形用户接口。开发环境包括 MATLAB 桌面及其命令窗口、命令纪录、帮助浏览器、工作平台、文件和搜索路径。MATLAB 6.0 与以前的版本相比，其工作环境更符合 Windows 的风格，采用多文档分割界面，使得 MATLAB 的编程更加轻松简单。

- MATLAB 数学函数库。该库收集了大量的从基本函数（例如求和、三角运算、复杂算术等）到复杂函数（例如矩阵求逆、求矩阵特征值、贝赛尔函数和快速福利叶变换等）的计算算法。
- MATLAB 语言。MATLAB 语言是一种包括流程控制语句、函数、数据结构、输入输出和面向对象编程方式的高级矩阵/数组语言，能够通过与其他 MATLAB 系统组成部分间的交互来完成非常复杂的计算任务。
- 图形句柄。图形句柄即 MATLAB 的图形系统，该系统既包括二维和三维数据可视化、图像处理、动画和图形描述等高级命令，又包括允许用户完全自定义图形并在 MATLAB 应用程序中建立完全图形用户界面的低级命令。
- MATLAB 应用程序接口（Application Program Interface API)。API 是允许用户编写 C 或 Fortran 与 MATLAB 接口程序的系统库，该库中包含一些调用工具，其他应用程序能够使用这些工具以动态链接、作为计算引擎、读写 MAT 文件三种形式调用 MATLAB 程序。

由于 MATLAB 具有使用简便、功能强大的优点，目前应用十分广泛，各个大学都将其作为数学、工程和科学专业学科的标准指导工具和高级课程来介绍；而在工业领域中，MATLAB 也是高生产力研究、发展和分析的有力工具。

1.2　MATLAB 6.0 开发环境概述

1.2.1　MATLAB 6.0 桌面概述

安装完成并启动 MATLAB 6.0 后，将看到如图 1-1 所示的 MATLAB 桌面（发布面板由于安装选项的不同可能略有不同），该桌面包括图形用户界面形式文件管理工具、变量和 MATLAB 有关应用程序。通过对工具界面进行打开、关闭、移动、改变大小等操作可以任意改变该桌面的外观。用户还可以将工具界面移出或拖回桌面。所有的工具界面都具备相同的外观特征，例如文本菜单和快捷键。如果希望改变工具界面的特征，可以通过选择 File 菜单的 Preferences 来实现。例如，可以通过该菜单来改变命令窗口的字体。

1.2.2　MATLAB 6.0 图形工具界面

从图 1-1 中可以看出，MATLAB 6.0 包含 6 种工具界面：

- 命令窗口（Command Windows）
- 发布平台（Launch Pad）
- 帮助浏览器（Help explorer）
- 当前目录浏览器（Current Directory）
- 工作平台浏览器（workspace）
- 编辑/调试器（Edit/Debug）

下面将向读者一一介绍。

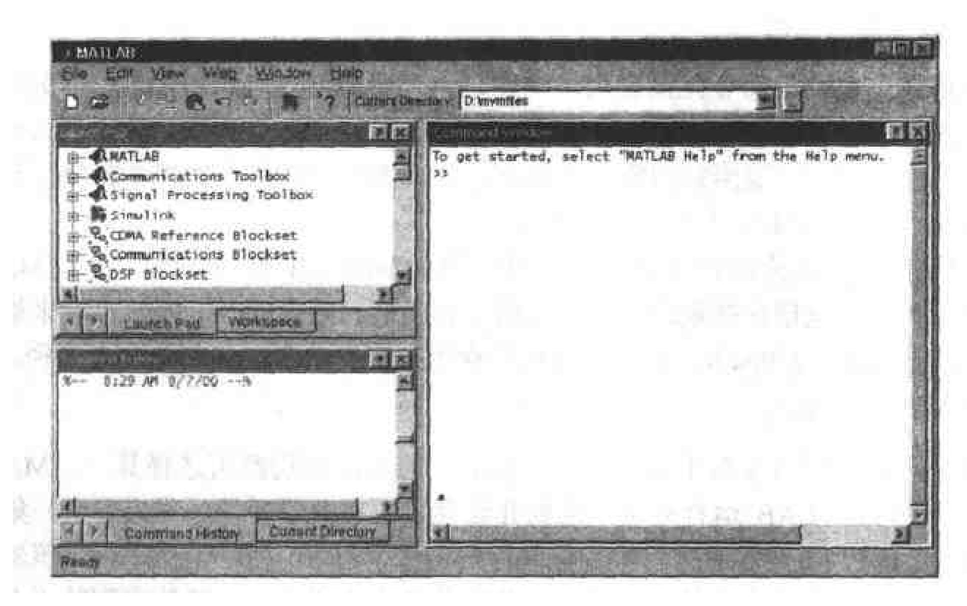


图 1-1　MATLAB 6.0 开发环境界面

1. 命令窗口

使用窗口来输入变量和运行函数及 M 文件。如果需要，用户还可以定义该窗口的输入输出特性，这一点将在以后的内容中予以介绍。

用户在命令窗口键入的命令行都记录在命令纪录（Command History）窗口中。在命令纪录窗口中，可以看到以前用到的函数，还可以通过拷贝来执行所选的命令行。如果需要将 MATLAB 运行期间的输入输出保存到一个文件中，可以使用函数 diary。

在 MATLAB 命令窗口中还可以运行外部程序。感叹号“!”作为脱离 MATLAB 外壳的提示符表示以下的输入命令均为操作系统命令，这对于那些不希望退出 MATLAB 程序而运行其他程序的用户是非常有用的。当用户退出外部执行程序后，操作系统自动将控制权交还给 MATLAB。

2. 发布平台

MATLAB 的发布平台为用户提供对工具箱、演示程序和文档的轻松访问，双击“工具箱”、“演示程序”和“帮助”等对相应的图标就可获得用户所需的操作。

3. 帮助浏览器

使用帮助浏览器可以搜索或查看用户所有 Mathworks 产品文档。帮助浏览器是一个集成在 MATLAB 桌面中的网络浏览器，显示 HTML 文档。点击图 1-1 中标注的问号图标或在命令窗口中键入 helpbrowser 命令即可进入帮助浏览器界面。

帮助浏览器包括两个窗体，一个是用于查询信息的帮助漫游器，另一个是浏览信息的显示窗体。帮助漫游器包括可通过标签切换的目录、索引、搜索、收藏夹四部分，同时提供产品过滤器以限定产品范围。显示面板则提供页面浏览、添加收藏夹、打印、和查找等功能。

另外，MATLAB 还提供一种在线帮助。如果在命令窗口键入 help 命令名或函数命令，窗口就将该命令或函数的功能介绍以及参数显示出来。本书中未作解释的函数建议读者用这种帮助方式获得详细信息。

4. 当前目录浏览器

当前目录浏览器与命令纪录公用一个窗口，通过标签切换。MATLAB 文件的操作使用当前目录并使用搜索路径作为参考点，用户希望运行的任何文件都必须位于当前路径或搜索路径内。当前目录浏览器用来搜索、查看、打开或修改 MATLAB 相关路径。另外，也可以通过函数 dir、cd 和 delete 来进行路径操作。

MATLAB 使用搜索路径寻找 M 文件和其他 MATLAB 相关文件，通常 MATLAB 提供的文件和工具箱都包括在搜索路径内。使用桌面菜单 File 中的 Set Path 选项来修改或添加搜索路径，函数 path，addpath，rmpath 也可分别用来查看、添加和删除搜索路径。

5. 工作平台浏览器

工作平台浏览器与发布平台共享一个界面，通过标签切换可选择其一。MATLAB 工作平台包括一些 MATLAB 运行过程中用到并储存于内存中的变量（称为数组）集合。用户通过使用函数、运行 M 文件或装载将变量保存到工作平台中。使用工作平台浏览器或函数 who 和 whos 来查看工作平台中的变量信息。使用菜单命令或 clear 函数来删除平台中的变量。MATLAB 运行结束后工作平台不被保持，如果希望在以后的 MATLAB 运行过程中使用该平台，可以使用菜单命令或函数 save 将平台保存为一个 MAT 文件，文件扩展名为.mat。读取 MAT 文件可以使用菜单或函数 load。

在工作平台浏览器中右击一个变量可以看到行编辑器，使用该编辑器可对工作平台的一维或二维常数数组、字符串或字符串数组元素进行编辑和查看。

6. 编辑/调试器

使用编辑/调试器创建和调试用户编写的旨在运行 MATLAB 函数的 M 文件。编辑/调试器提供基本文本编辑和 M 文件调试的图形用户界面。用户也可以使用其他文本器来创建 M 文件，并允许使用桌面 File 菜单中的 Preferences 将该编辑器设为默认的编辑器。如果用户使用其他文本编辑器，仍然可以使用 MATLAB 编辑/调试器对 M 文件进行调试。另外还可以使用调试函数进行调试，例如使用 dbstop 设置断点。如果用户只是希望查看一个 M 文件，可以在命令窗口中键入 type 函数实现。

1.2.3　开发环境其他特征

MATLAB 6.0 还具有以下特征：

- 数据导入导出——将其他应用程序创建的数据导入 MATLAB 工作平台的技术。这包括导入向导和 MATLAB 工作平台变量打包以供其他应用程序使用。
- 提高 M 文件执行效率——提供一个看起来是 M 文件运行时间测量工具，使用它可以使用户提高执行速度。
- 源控制程序接口——在 MATLAB、Simulink 和 Stateflow 中访问用户的源控制系统系。
- 使用笔记本——在一个字处理器环境中（Word）访问 MATLAB 的数字计算和可视化软件。

1.3 MATLAB 矩阵基本操作

1.3.1 矩阵

在 MATLAB 中，矩阵是一个数值数组，下标从 1 开始。单个数值可以认为是一个 1×1 的矩阵向量，也可以认为是一个标量。MATLAB 采用不同的存储方式来存储数值和非数值数据。一般的程序语言一次仅处理一个数值，但 MATLAB 将矩阵操作设计得非常自然，允许用户对整个矩阵进行迅速简便的操作。

将矩阵输入 MATLAB 可以采用以下几种方法：

- 输入矩阵元素准确列表
- 从外部数据文件中装入数据列表
- 使用 MATLAB 函数产生矩阵
- 在 M 文件中采用用户自定义的函数生成矩阵

在命令窗口中执行第一种矩阵输入方式要遵循以下原则：整个矩阵以方括号开始和结束，同行元素间使用逗号或空格分开，一行元素以分号结束。用户输入的矩阵将自动保存到 MATLAB 工作平台中，用户可以使用矩阵名在工作平台中访问该矩阵。

第二种方法是使用 load 命令读取二进制或文本矩阵数据文件，该文件每行包括矩阵的一行元素，每行元素间以空格分开。假设文件名为 matrix.dat，命令 load matirx.dat 将创建一个名为 matrix 的矩阵。使用导入向导读取文本或二进制文件更为简单。

MATLAB 提供四个产生基本矩阵的函数：

全零矩阵 zeros；

ones：全 1 矩阵；

rand：均一分布随机矩阵；

randn：一般随机矩阵，可以用这四个函数创建矩阵。

使用 MATLAB 编辑器或其他文本编辑器创建 M 文件，用户可以在该文件中定义矩阵，定义原则与第一种方法相同。M 文件成功保存后可以作为函数使用，运行该函数即可得到文件中定义的矩阵。

如果希望提取矩阵 A 第 i 行 j 列的元素，可以使用语句 A(i,j)，i、j 可以大于矩阵的维数，这相当于给矩阵添加行或列。如果希望访问多个数据，还可以冒号操作符 “:” 来完成。冒号操作符是 MATLAB 最为重要的操作符之一，其表达式如下：

a:±b:c

上式表示从 a 到 c，按步长 b 变化的一个行向量，如果 b 为正，则须 a 小于 c，行向量数值递增，否则 a 大于 c，行向量数值递减。b 可缺省，缺省值为 1。这样，如果要访问矩阵 A 从 1 到 k 行的第 j 个元素值，可用表达式 A(1:k,j)；如果要删除某列元素，则只需执行语句：

A(:,j)=[ ]

另外，MATLAB 还提供矩阵合并功能。假设 A 是 4×4 矩阵，语句：

B = [A A+32;A+48 A+16]

则生成一个 8×8 矩阵 B。

1.3.2　表达式

一个表达式通常由以下几个部分构成：

- 变量
- 常数
- 运算符
- 函数

前面提到过，MATLAB 是一种基于不限维数组的语言，由于语言的机制不同，因而 MATLAB 的表达式与其他编程语言的表达式也有所不同。

MATLAB 语言不需要任何数据类型声明或维数描述，当 MATLAB 遇到一个以字符串表示的新变量名时，系统自动创建该变量并给该变量分配相应的存储空间。如果该变量已经存在，MATLAB 将根据需要修改它的内容，如果有必要也会重新分配存储空间。变量名由一个字母开头，后面可以跟字母、数字或下划线，MATLAB 将使用前 31 个字符作为变量名。MATLAB 区分大小写，a 和 A 并非同一个变量。

MATLAB 采用十进制数，常数的虚部用 i 或 j 来表示，所有的数值都使用双精度浮点数标准格式存储，取值范围从 10^{308} 到 10^{-308}。

MATLAB 采用的算术运算符和优先级与其他应用程序基本一致，只是增加了一些专门用于矩阵处理的运算符，有关这些运算符将在以后的内容中予以介绍。

MATLAB 提供大量的标准基本数学函数，例如：绝对值求取函数 abs，开方函数 sqrt，指数函数 exp，正弦函数 sin 等。如果对一个负数进行开方或求对数，函数自动输出近似的负数结果。MATLAB 还提供许多高级数学函数，例如贝赛尔函数等，大多数函数接受复型参数。有一部分函数，例如 sqrt 和 sin 函数，是 MATLAB 内置函数，这些函数是 MATLAB 内核的一部分，因而非常有效，但是函数的计算过程是不可知的。另外一些函数，例如 sinh，则是以 M 文件形式给出的，本身也是一个 MATLAB 应用程序，因而用户可以访问并修改函数的代码。有一些函数能够产生一些非常有用的常数，例如：

- pi：π，3.14159265……
- inf：无穷大
- NaN：非常数

1.3.3　深入矩阵和数组操作

根据前面的介绍可以知道，MATLAB 非常擅长于矩阵和数组的操作，这可以集中体现在以下三个内容中。

1. 线性代数

准确说来，矩阵就是一个代表某种线性变换的二维数组，而 MATLAB 又是以数组为基本单位的语言，因而 MATLAB 非常擅长于线性代数这门专门研究矩阵数学计算的学科。

MATLAB 除了可以对整个矩阵进行简单的线性运算，例如加减乘除，还可以进行一些特殊的代数运算，例如矩阵特征值、特征向量的求取。

矩阵的加减乘运算都比较简单，除法有点特殊，分为左除和右除两种情况，假设有表达式：

X=A\B (1)

和 X=A/B (2)

（1）是矩阵右除（即通常的除法）表示 X 是矩阵方程 AX=B 的解，而（2）则是矩阵左除，表示 X 是矩阵方程 XA=B 的解。矩阵的其他基本运算还有幂运算"^"、转置运算"'"等等。

矩阵的一些特殊运算基本上都是由 MATLAB 的内核函数完成的。矩阵运算函数非常丰富，常用的有以下几种：

矩阵求秩：det

矩阵求逆：inv

矩阵特征值求取：eig

矩阵特征多项式系数求取：poly

2. 数组

矩阵和数组的加减运算是相同的，但是矩阵乘除法和数组的乘除法操作是不同的。MATLAB 乘法使用点号"."作为数组乘法的运算符之一，如果在运算符"*"、"/"或"\"前加一个点号，则表示两个数组的数据将一一进行运算，而不是像矩阵那样做矩阵乘法、左除或右除。数组的这种操作在编写建表过程中非常有用，同样地，数组的基本数学函数对数组的操作也是以元素为单位进行的。

例如：

A=[2 3 4]

则： A.^2=[4 9 16]

而 A^2 操作是不允许的，因为 A 并非方阵。

3. 多变量数据

MATLAB 使用列定位方法对多变量统计数据进行分析。数据集的每一列代表一个变量，每一行代表一个观测，元素(i,j)表示第 i 个观测的第 j 个变量。用户可以使用 MATLAB 数据分析函数来分析以上意义的数据集，例如求数据集均值的函数 mean，标准方差函数 std 等等。

除了以上的优越的矩阵和数组操作方法，MATLAB 还提供以下几种非常有用的操作：

- 标量扩展，例如矩阵与标量的加减法运算等；
- 逻辑下标，例如以一个矩阵 a 为下标访问另一个矩阵 b 等；
- 搜索函数，例如用根据一个表达式的运算结果来访问矩阵的某个元素等。

1.3.4 命令窗口输入输出控制

用户已经实践过使用 MATLAB 命令行键入命令和表达式，以及观察命令窗口输出的结果，而 MATLAB 还提供一些方法来控制命令窗口的输入输出，这些方法包括：

- 控制输出值的外观；
- 使用 MATLAB 命令禁止输出；

● 在命令行中键入长命令；
● 编辑命令行。

下面将一一介绍这些方法。

1. format 命令

format 命令控制 MATLAB 显示数据的格式。该命令仅仅影响数值的显示，不影响 MATLAB 对数值的计算和存储。常用的格式有：

short，short e，short g，long，long e，long g，bank，rat，hex，compact。

例如假设：x = [4/3 1.2345e –6]

若运行语句：format short

则运行结果为：1.3333 0.0000

如果用户希望对输出进行进一步的控制，使用 sprintf 和 fprintf 函数。

2. 禁止输出

如果用户键入一个语句后直接回车，MATLAB 将自动在屏幕上显示该语句的运行结果。如果用户使用分号结束该行，则 MATLAB 仅执行计算而不显示任何输出。这对于用户生成一个大矩阵而言尤为重要。另外，以百分号“%”开头的行为注释行，不产生任何输出结果。

3. 键入长命令

如果一个语句长于一个命令行，在该行回车前使用三个点“...”表示该语句将在下一行中继续。

4. 命令行编辑

用户键盘上不同的箭头和控制键允许用户进行查询、编辑和重用用户以前键入的命令。例如，如果用户错误地键入：

rho = (1 + sqt(5))/2

其中 sqt 拼写错误，应为 sqrt，MATLAB 会做出这样的反应：

Undefined function or variable ‘sqt’

用户无需重新键入整个行，简单使用箭头“↑”，错误行将重新显示。使用箭头“←”将光标移动到相应地点插入丢失的字符 r 即可得到正确的结果。“↑”表示重新调用以前的行，如果先键入一个新字符再使用“↑”则会找到以前以这个字符开头的行。用户也可以从命令行纪录窗口中拷贝执行命令。不同的计算机可用的命令行编辑键也不同，用户可以在自己的机器上试验性地使用。

1.4 实 例 讲 解

【实例一】：使用 MATLAB 产生一个 4×4 的魔方阵，并对该矩阵进行验证。

分析：用户将在这个问题的解决过程中初步掌握如何利用命令窗口完成简单的计算工作。MATLAB 提供一个特殊的矩阵函数 magic，使用该函数能够生成任意大小的魔方阵，

该阵的行、列和对角线之和相等。

解决方法：

步骤一：启动 MATLAB 6.0 软件；

步骤二：在 MATLAB 6.0 命令窗口中输入以下语句：

```
a = magic(4)
```

步骤三（系统自动完成）：结果显示：

```
a =
     16   3   2   13
     5   10   11   8
     9    6   7   12
          4   15   14   1
```

步骤四：验证矩阵列之和相等。函数 sum 能够计算矩阵各列之和，使用这个函数来验证该魔方阵的各列之和相等。输入命令：

```
sum(A)
```

计算结果：

```
ans =
  34   34   34   34
```

验证正确。

步骤五：验证矩阵行之和相等，为了使用函数 sum 求出矩阵 a 的各行之和，首先要对矩阵进行转置，然后再求各列之和。输入语句：

```
sum(A')
```

计算结果：

```
ans =
34   34   34   34
```

验证正确。

步骤六：验证对角线之和正确。采用函数 diag 提取矩阵 a 的对角向量，然后用 sum 读该向量求和。输入语句：

```
sum(diag(A))
```

计算结果：

```
ans =
 34
```

步骤七：验证反对角线之和。由于反对角线并不常用，所以 MATLAB 没有提供直接提取反对角线的函数，所以先使用函数 fliplr 将矩阵 a 水平翻转，然后提取对角线向量求和。输入语句：

```
sum(diag(fliplr(A)))
```

计算结果：

```
ans =
34
```

验证正确。

【实例二】：导入格式如下的磁盘数据文件 data.dat **到** MATLAB **中：**

```
16.0   3.0    2.0   13.0
5.0   10.0   11.0    8.0
9.0    6.0    7.0   12.0
4.0   15.0   14.0    1.0
```

分析：这个问题有两种解决方案：一是利用 MATLAB 提供的数据导入向导，二是利用 MATLAB 命令。第一种方法比较简单，这里不作介绍。使用第二种方法要注意，如果数据文件不在 MATLAB 的当前路径或搜索路径中，则 MATLAB 将找不到该文件，因而也无法导入数据。为此，需要先将数据文件拷贝到 MATLAB 的当前或搜索路径中，再进行数据装载，然后开始计算。

解决方法：

步骤一：查询 MATLAB 的当前路径。输入语句：

```
path
```

MATLAB 将在命令窗口中显示其所有的当前搜索路径。用户可以观察数据文件是否在这些目录下，如果不在，则必须执行步骤二进行数据文件拷贝。这一步骤亦可以省略。

步骤二：将数据文件 data.dat 拷贝到某当前搜索路径中。这里假设用户是在 windows 环境下编程，MATLAB 软件位于 d:\MATLAB 下。为了拷贝数据文件，MATLAB 将控制权先交给操作系统完成文件拷贝。输入语句：

```
!copy   c:\data.dat   d:\MATLAB\bin
```

步骤三：导入数据文件到 Maltab 中。输入语句：

```
load data.dat
```

步骤四：检验导入的数据是否正确。输入语句：

```
data
```

显示结果：

```
data =
        16.0   3.0    2.0   13.0
         5.0  10.0   11.0    8.0
         9.0   6.0    7.0   12.0
         4.0  15.0   14.0    1.0
```

导入正确。

1.5 小　　结

通过以上的介绍和实例演示，用户可能已经初步体会到了 MATLAB 的强大功能。MATLAB 对矩阵和向量操作的便利性使得用户能够轻松掌握诸如系统控制、图像处理等矢量操作。

在本章中，首先对 MATLAB 系统的概念和特点进行了简要介绍，并对 MATLAB 系统

的五个组成部分一一进行了说明，使用户将来能够充分地利用 MATLAB 进行编程。

MATLAB 6.0 开发环境与 MATLAB 以前的版本有较大区别，尤其是工作桌面的外观。MATLAB 6.0 改变了以前单一的命令窗口，整个工作桌面包括命令窗口、命令记录、发布平台、当前路径浏览器、帮助浏览器和编辑/调试器这六个图形界面，使得用户对 MATLAB 的操作更加便利。

MATLAB 语言与其他语言相比的最大特点在于：MATLAB 是建立在矢量基础上，以数组为基本数据类型的。本章的内容，力求使用户掌握 MATLAB 基本矩阵操作，为今后的深入编程打下基础。

本章的内容仅仅体现了 MATLAB 系统庞大功能的一小部分，在以后的学习中，用户可以学到 MATLAB 其他强大之处，一个最好的例子就是 MATLAB 的图形系统。

第 2 章　MATLAB 程序设计精要

在第一章的基础上，读者将在本章中学会如何编写较为高级的 MATLAB 应用程序。读者将了解 MATLAB 的基本程序控制结构以及 MATLAB 四种特殊数据类型，同时掌握 MATLAB 的脚本、函数两种 M 文件的编写方法并能够利用一些提高程序执行效率的编程方法。另外，本章还将使读者初步掌握 MATLAB 与其他编程软件的交互使用方法。

2.1　MATLAB 流程控制

2.1.1　MATLAB 编程简介

MATLAB 本质上是一种解释性的语言，也就是说，MATLAB 是通过逐行扫描方式对程序进行编译执行的。MATLAB 的基本编程方法有两种，一种是直接在命令窗口下的提示符后键入用户需要执行的语句或命令，另外一种是在 MATLAB 编辑/调试器界面下编写、存储和运行 MATLAB 应用程序——M 文件。

第一种方法适用于程序比较简单的情况。在命令窗口下键入操作命令或函数后可以直观地看到输出结果，适合作为机器演草纸。但这种方法有一个很大的缺点就是不适用于程序的反复调试和代码修改。

第二种方法是用户开发 MATLAB 程序较好的编程手段，用户可以利用编辑/调试器的强大功能对自己编写的 M 文件进行调试和修改。在 File 菜单下选择产生新的 M 文件，即可进入编辑/调试器界面开始编程。

从编程思想的角度而言，MATLAB 与其他高级语言的主要区别在于：MATLAB 的语言结构非常简单，编程的主要内容集中在系统内核函数或工具箱函数的使用上，而其他软件则更强调流程控制的力度。正是由于这个原因，MATLAB 的流程控制语句比较简单，主要有以下 4 种：

- 假设语句：if
- 条件转移语句：switch 和 case
- 循环语句：for 和 while
- 循环结束语句：continue 和 break

1. 假设语句 if

if 语句首先对给出的逻辑表达式进行求值，如果该表达式的值为真（true）则执行指定的语句群，可选的关键字 elseif 和 else 可以用来执行表达式为假时需要执行的语句群。与 if 相匹配的结束关键字是 end，end 必须是语句群的最后一个语句。这四个关键字都不包含方

括号或小括号。例如：

```
if rem(n,2) ~= 0                    %求n除以2的余数
    M =odd_ magic(n)                %若n为奇数则生成一个
elseif rem(n,4) ~= 0                %求n除以4的余数
    M = single_even_magic(n)        %若n不为4的整倍数则生成n维单偶魔方阵
else
    M = double_even_magic(n)        %若n为4的整倍数则生成n维双偶魔方阵
    end
```

有几个将矩阵比较结果转化为布尔量的函数对这种假设控制程序块非常有用，常用的函数包括：

- 矩阵相等比较函数：isequal；
- 空矩阵检查函数：isempty；
- 矩阵元素是否全为非零函数：all；
- 矩阵是否有非零元素检查函数：any。

2. 条件转移语句 switch 和 case

switch 语句根据一个变量或表达式的具体取值情况来执行相应的语句群。语句群由 case 或 otherwise 组成，如果该变量或表达式符合 case 的条件则执行 case 所指定的语句群。switch 结构必须由 end 来结束。这里要注意的是，与 C 语言的 switch 不同，MATLAB 的 switch 不是对所有的 case 表达式都进行比较，如果第一个 case 语句为真，其他 case 语句就不会执行，所以无需 break 语句。例如：

```
switch (rem(n,4)= =0) + (rem(n,2)= =0)
case 0
    M = odd_magic(n)
case 1
    M = single_even_magic(n)
case 2
    M = double_even_magic(n)
otherwise
   error('This is impossible')
end
```

3. 循环语句 for 和 while

for 循环按照事先指定的初始值和固定的循环次数重复执行一个语句群，以 end 结束该语句群。例如：

```
for i = 1:m
   for j = 1:n
         H(i,j) = 1/(i+j);
   end
end
```

与 for 循环不同的是，while 循环是通过控制一个逻辑状况来对一个不限定次数的语句

群重复执行过程，end 同样作为语句群结束语句。例如：

```
    a = 0 ;   fa = -Inf ;          %Inf表示无穷大
    b = 3;   fb = Inf ;
    while b-a > eps*b              %eps表示浮点数的精度
    x = (a+b)/2;
    fx = x^3-2*x-5;
    if sign(fx) == sign(fa)        %sign为符号求取函数，sign(x)=x/|x|
        a = x; fa = fx;
    else
        b = x; fb = fx;
    end
end
```

4. 循环控制语句 continue 和 break

continue 语句控制其所在的 for 和 while 循环，使之忽略 continue 以下的其他循环体语句而进入下一次循环。例如：

```
    fid = fopen('magic.m','r');      %fopen为文件打开函数，返回值为文件标示符
    count = 0;
while ~feof(fid)                     %feof: 查询文件是否有结束标示符
    line = fgetl(fid);               %fgetl: 获取文件下一行内容
    if isempty(line) | strncmp(line,'%',1)    %isempty: 检查矩阵是否为空
                                              % strncmp: 字符串比较函数
        continue
    end
    count = count + 1;
end
```

break 语句可以使用户提前从 for 或 while 循环中退出，例如：

```
    a = 0; fa = -Inf;
    b = 3; fb = Inf;
    while b-a > eps*b
    x = (a+b)/2;
    fx = x^3-2*x-5;
    if fx == 0
        break
    elseif sign(fx) == sign(fa)
        a = x; fa = fx;
    else
        b = x; fb = fx;
    end
end
```

这里要注意，在嵌套循环中，break将仅从内层循环中退出。

2.2 深入MATLAB编程

2.2.1 MATLAB其他数据类型

除了基本的数据类型矩阵和数组以外，MATLAB还支持以下四种较为特殊的数据类型：

- 多维数组
- 单元数组
- 字符串和文本
- 结构体

下面将对这几种数据类型进行详细介绍。

1. 多维数组

MATLAB 中的多维数组是指下标在二维以上的数组。多维数组可以通过调用输入参数在两个以上的函数zeros，ones，rand或randn来生成，例如：

```
R = randn(3,4,5);
```

一个三维数组也许代表着一组有实际意义的三维物理数据，例如按照方形网格采样的房间温度，或者代表一个矩阵序列 A^K，或者代表与时间有关的矩阵 A(t)。多维数组的访问和计算方式与二维数组类似，例如表达式A(i,j,k)即表示第k个矩阵的第i行第j个元素，同样也可以使用冒号操作符来访问矩阵数据，例如：

```
    M = zeros(4,4,24);
    for k = 1:24
    M(:,:,k) = k;
end
```

2. 单元数组

MATLAB 的单元数组实际上是一种其元素是其他数组拷贝的多维数组，它与多维数组的主要不同在于：三维数组存储的是相同大小的矩阵序列，而单元数组可以用来存储不同大小的矩阵序列，例如：

```
C = {A   sum(A)   prod(prod(A))}          %prod: 求矩阵各列乘积
```

虽然一个空矩阵的单元数组可以通过cell函数创建，但一般情况下单元数组都是由大括号“{}”括起来的复杂元素集合，每一个元素都可以是二维以下的数组，一个元素称为一个单元。这里有两点非常重要：一是访问单元数组的一个单元是通过使用含下标的大括号来完成的，例如C{1}将得到单元数组C的第一个元素；二是单元数组包括的是其他数组的备份，而不是这些数组的指针，如果改变某一个单元（元素）的值，对该单元数组不会有什么影响。

3. 字符和文本

使用单引号“'”在MATLAB中输入文本，将会产生一个字符数组，而不是前面所讲的

数值数组。例如：

s = 'Hello'

在系统内部，字符数组仍按照数值（ASCII 码）存储，但并非数值数组的双精度浮点数存储格式。也可以使用类型转换函数将字符或文本数组转化为按浮点数存储格式存储的数值矩阵，例如以下语句可将上例中的字符数组 s 转化为数值矩阵 a，a 的每一个元素都是浮点数，代表相应字符 ASCII 码：

```
a = double(s)
a =
    104    101    108    108    111
```

字符串矩阵也可以像数值矩阵一样进行赋值、合并等操作。如果对一个包括不同长度行的字符串数组进行操作，用户可以有两种选择：一是将该数组各元素填补为相同长度，二是使用单元字符串，单元字符串的概念同单元数组概念相同，在此不作赘述。

4. 结构体

MATLAB 的结构体实际上是一种通过文本域指示器访问元素的多维数组，例如定义：

```
s.name = ' Ed plum '
s.score = 83
    s.grade = 'B+'
```

由于结构体也可以是一个数组，所以用户可以在结构体中插入其他元素，此时结构体数组的每一个元素都是一个包含几个域的结构体。例如：

```
s(2)= struct('name','Jerry','score',70,'grade','C')
```

另外，用户可以单独访问每一个元素的某个域的内容，也可以同时访问整个结构体某个域的内容。下述两个语句的执行结果是相同的：

```
[s. score ]
[s(1).score,s(2).score,s(3).score]
```

2.2.2 脚本与函数

包含 MATLAB 语言代码的文件称为 M 文件，即 MATLAB 应用程序。M 文件分为两种：

- 脚本：脚本没有输入输出参数，仅在 MATLAB 工作平台中进行数据操作。
- 函数：能够接受输入参数并返回输出参数，函数中使用内部变量。

开始使用 M 文件时，用户一般在当前目录下进行 M 文件的编写，但随着编程程度的加深，用户可能会希望在其他路径中组织自己的 M 文件以及可添加到 MATLAB 搜索路径中的工具箱，此时要注意如果用户的函数名发生重复，MATLAB 执行在搜索路径中先搜索到的函数。

1. 脚本

当用户调用脚本时，MATLAB 将简单地执行该文件中的命令。脚本可以对工作平台中已存在的数据进行操作，也可以创建新的操作数据。尽管脚本并不返回输出参数，但脚本创建的所有变量都保存在工作平台中以备以后的计算应用。另外，脚本也可以使用图形函数生成图形。假设有一个名为 magicrank.m 的脚本文件，代码如下：

```
    r = zeros(1,32);
    for n = 3:32
    r(n) = rank(magic(n));      %rank：矩阵求秩函数
end
    r
bar(r)                          %bar：直方图绘制函数
```

在命令窗口中执行该函数：

magicrank

变量 n 和 r 将保存在工作平台中，同时输出如图 2-1 所示的图形结果。

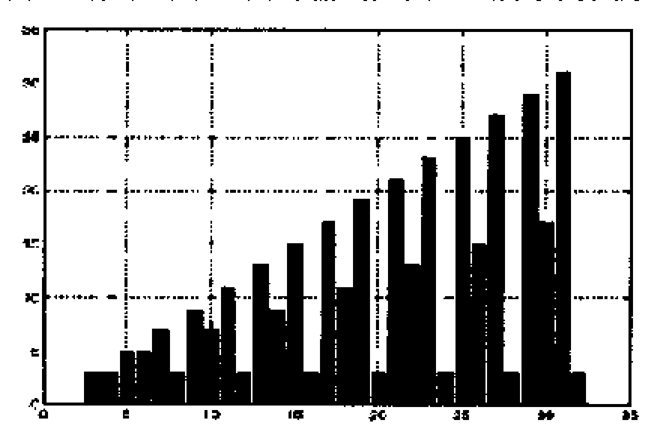


图 2-1　脚本文件 magicrank 图形结果

2. 函数

函数是可以接受输入参数并返回输出参数的 M 文件，函数的定义结构如下（注意函数名与 M 文件名必须一致）：function　输出参数=文件名(输入参数)

调用格式为：返回值=函数名(输入参数)

例如：

```
    function r=rank(A,tol)
    s = svd(A);                  %svd：奇异值分解函数
    if nargin= =1                %nargin：函数输入参数个数
        tol = max(size(A)') * max(s) * eps;    %max：求矩阵最大元素的函数
    end
r = sum(s > tol);
```

另外，当输入参数为字符串时，还可以使用以下调用方式：

函数名　输入参数 1　输入参数 2　……

但是要注意，这种调用方式不能得到函数的返回值，而且如果输入参数不是字符串，这种调用方法就会产生 MATLAB 无法识别的错误，导致错误的计算结果。例如：

A = [2，3；4，5]

执行：eig A

这时，MATLAB 不会报错，但会产生如下的错误结果：

ans = 65

这是由于 MATLAB 将字符 A 的 ASCII 码（65）理解为 eig 函数的输入求得特征值为 65。

函数对变量的操作是在自身的而不是 MATLAB 的工作平台中进行的，因而局部变量不可在各函数间共享。如果用户希望多个函数能够访问同一个变量，只需将该变量在每一个函数中利用关键字 global 定义为全局变量，注意全局变量的声明必须在该变量真正使用之前。例如：

```
function h=falling(t)
global GRAVITY
h=1/2*GRAVITY*t.^2;
```

如果一个全局变量发生变化，在所有函数中该变量都会变化，因而用户无需修改其他函数就可以得到新的结果。

另外，用户还可以使用 eval 函数将文本作为输入参数以执行文本宏命令，这在使用 MATLAB 应用程序接口与其他程序进行交互时非常有用。表达式或语句 eval（s）使用 MATLAB 编译器对用 s 表示的表达式进行求值或执行 s 语句。例如：

```
s='magic(4,4)';
eval(s)
```

与 magic(4,4)执行结果相同。

2.2.3 矢量化方法

为了获得高效的 MATLAB 程序，对用户的 M 文件进行矢量化是非常重要的，例如，其他程序语言使用 for 或 do 循环完成某项工作时，MATLAB 可以使用向量或矩阵操作来完成，程序执行速度将会大大提高。例如如果需要创建一个对数表，可以使用以下程序：

```
x=.01
for k=1:1001
    y(k)=log10(x);       %log10: 对数求取函数
    x=x+.01;
end
```

而相同代码的矢量化版本如下：

```
x=.01:.01:10
y=log10(x);
```

显然，矢量化后的程序代码执行效率大大提高。虽然当程序代码较为复杂时，矢量化的作用不是非常明显，但是如果运行速度是程序运行的重要指标时，用户应该尽量将自己的算法矢量化。

2.2.4 预分配方法

如果无法实现对某段代码的矢量化，用户还可以通过为所有输出向量或数组预分配存储空间来提高 for 循环的执行速度。例如以下代码利用函数 zeros 对 for 循环中创建的向量进行预分配，使得 for 循环的执行速度显著提高：

```
r=zeros(32,1);
```

```
for n=1:32
    f(n)=rank(magic(n));
end
```

如果没有对向量进行预分配，MATLAB 编译器将在每次循环中扩大一次 r 向量，而通过预分配则可省略这一步骤，使程序执行速度提高。

2.2.5 函数句柄

用户可以为每一个 MATLAB 函数创建一个句柄并将这个句柄作为该函数的参考方式来使用。函数句柄将一个参数表传递给其他函数，其他函数可以利用句柄进行求值运算或执行某种命令。在函数名前使用记号“@”就可以构造一个函数的句柄，该句柄可以赋给一个变量，例如，下例中使用函数 plot-fhandle 接受函数 sin 的句柄，并使用 feval 函数利用该句柄进行求值。

```
plot-handle(@sin,-pi:0.01:pi)
```

该语句的执行结果将是一个周期的正弦波。函数 plot-handle 定义如下：

```
function x=plot-handle(fhandle,data)
  plot(data,feval(fhandle,data))
```

2.2.6 功能函数

对标量变量的非线性函数进行操作的一类函数叫做功能函数。功能函数包括以下几种：

- 找零函数
- 最优化
- 求积函数（定积分的近似解法）
- 普通微分方程

使用 MATLAB 功能函数要注意一点，功能函数并不是直接对非线性计算函数进行运算，而是利用该函数的句柄进行计算。

2.3 MATLAB 与其他应用程序接口

2.3.1 API 概述

尽管 MATLAB 开发环境能够完全满足程序开发和数据处理工作，但在有些情况下，与 MATLAB 外部程序进行数据和程序交互是非常有用的，MATLAB 提供一个应用程序接口（API）来支持这种外部交互。API 支持的功能包括：

- 在 MATLAB 中调用 C 或 Fortran 程序；
- 在 MATLAB 环境中导出或导入数据；
- 建立 MATLAB 和其他软件程序间的用户/服务器关系。

第一种功能主要是通过 MEX 文件来实现的。用户可以在 MATLAB 中像调用 MATLAB 内置函数一样来调用自身的 C 或 Fortran 程序，MATLAB 称这种 C 或 Fortran 程序为 MEX 文件。MEX 文件是一种 MATLAB 编译器，可以自动装载并执行的动态链接文件。MEX 文件在以下两种情况下非常适用：

- 对于用户已经编写好的大型 C 或 Fortran 程序，无需重新编写代码就可以在 MATLAB 下直接调用；
- MATLAB 的某些运行速度较慢的瓶颈计算（通常是 for 循环）可以被 C 或 Fortran 有效地重新编码执行。

使用 MEX 文件也要注意，如果用户编写的 C 或 Fortran 程序非常耗时或编程目的是为了执行低级指令操作，那么 MEX 文件的使用反而会影响 MATLAB 程序的效率。

API 的第二种功能主要是通过调用 MAT 文件实现的。MATLAB 环境下数据的导入导出有很多种方法，但 MAT 文件使用方法是其中最重要的一种。MAT 文件实际上是一种以 MATLAB 文件磁盘存储格式存储的文件，可以在各个不同的操作平台或不同单机 MATLAB 环境间进行数据交互。为了方便用户对 MAT 文件的操作和使用，MATLAB 提供一个以 mat 为前缀的可访问子程序库来实现 C 或 Fortran 程序对 MAT 文件的读写。MAT 文件的操作方法与具体的操作系统有关。

API 支持的第三种功能实际上是指 MATLAB 计算引擎的使用。MATLAB 引擎库是一个允许用户在自身的程序中调用 MATLAB 的子程序库，用户程序因而将 MATLAB 作为其计算引擎来使用。MATLAB 引擎操作作为一个单独的线程在用户程序后台运行，因此用户程序在执行时无需将 MATLAB 全部链接到用户程序中，只要调用其中的一小部分引擎通信库就足够了。在 Windows 系统下 MATLAB 引擎库的使用是通过 ActiveX 技术来实现的。

本节内容均以 Windows 系统为例进行说明。

2.3.2　MEX 文件的使用方法

在开始 MEX 文件的正式使用之前，用户还必须了解 MATLAB 是如何向其他语言描述其所支持的数据类型的。MATLAB 仅使用一种单一的对象类型——MATLAB 数组——来描述各种数据，所有的向量、矩阵、多维数组、单元数组、文本和结构体都作为 MATLAB 数组来存储，统称为 mxArray。C 语言在使用 mxArray 时要求其包括以下几个部分内容：

- MATLAB 变量名
- 维数
- 类型
- 变量是实数还是复数
- 变量是否为稀疏矩阵

只有定义了以上内容，C 语言才能够为 MATLAB 进行正确的数据处理。

MEX 文件的具体使用要通过以下步骤来实现：

（1）在使用 MEX 文件之前，必须为用户所使用的编译器组态。假设用户使用软件 VC 进行 C 语言编程。在 MATLAB 命令窗口或 DOS 下输入语句：

```
mex _setup
```

操作系统将会将其所支持的 C 编译器列举出来，选择 VC 及其正确的版本，然后输入

该编译器的具体路径，确认后系统设置就完成了。

（2）为了测试编译器组态是否成功或正确，最好试运行一个 MATLAB 提供的例程，这些例程全部位于 MATLAB 安装路径的 extern\example\mex 子路径下。假设选择一个名为 yprime.c 的文件，输入语句：

mex yprime.c

这将为 yprime.c 相应的 MEX 文件创建一个.dll 扩展名（具体的扩展名与具体的操作系统有关）。然后就可以在 MATLAB 中像调用 M 文件一样调用 yprime，输入语句：

yprime(1,1:4)

如果显示结果如下，则表明组态成功。

```
ans =
    2.0000 8.9685 4.0000 -1.0947
```

（3）下一步要编写 MEX 文件的 C 源代码。MEX 文件包括两个不同部分：

- 计算子程序。计算子程序中包括用户需要在 MEX 文件中实现的计算任务（包括数据输入输出）的源代码；
- 门关子程序。门关子程序是计算子程序与 MATLAB 的接口函数，主函数为 mexFunction 函数以及该函数的参数 prhs、nrhs、plhs 和 nlhs。门关子程序将计算子程序作为其子函数进行调用。

在门关子程序中，用户可以访问 mxArray 结构的数据，并在计算子程序中对该数据进行操作。用户在门关子程序中调用计算子程序后将设置一个 mxArray 类型的指针给门关子程序的返回值，Matalb 根据这个指针从用户的计算子程序中辨认输出数据作为 MEX 文件的输出。

MEX 文件的计算子程序和门关子程序两部分可以分开也可以在同一个文件中。无论哪种情况，源代码文件头必须包含以下链接来定义主函数和接口路径：

```
#include "mex.h"
```

门关子程序必须是主函数 mexFunction 并包含以下参数：

```
void mexFunction(
        int nlhs, mxArray *plhs[],
          int nrhs, const mxArray *prhs[])
          /* 其他 C 源代码*/
```

参数 nlhs 可以视为计算子程序调用的输出参数（称为左手参数），nrhs 视为输入参数（右手参数）。参数 plhs 是 MEX 文件的输出参数指针向量，prhs 则是输入参数指针向量。例如，如果在 MATLAB 下调用 MEX 文件 fun：

x=fun(y,z)

则 nlhs=1，nrhs=2，plhs 是一个空指针，prhs 包含两个分别指向 y，z 的指针向量。plhs 之所以是一个空指针是因为在门关子程序未执行之前不会创建输出变量 x。门关子程序负责创建一个输出数组并在 plhs[0]中指定这个数据的指针。

MEX 文件代码编写完成后，按照以上介绍的步骤对该文件进行编译连接：

mex timestwo.c

现在就可以调用该函数进行计算了：

```
x = 2;
y = timestwo(x)
y =
    4
```

2.3.3　MAT 文件的使用方法

MAT 文件为用户提供了从 MATLAB 中导入或导出数据的方法。这里首先要说明的一点是，虽然 M 文件和 MAT 文件都能够在不同机器间直接传输，但这两种文件的传输机制是不同的。M 文件能够传输是因为 M 文件是一种与操作平台无关的文件类型，而 MAT 文件能够传输则是因为该文件的头信息中包含了机器信息，MATLAB 在调用 MAT 文件时首先查询机器信息，如果与本机不符则自动进行转换。

MAT 文件的使用步骤与 MEX 文件类似，这里仅介绍 MAT 文件的编译连接方法和 MAT 文件编写的方法。仍以 VC 为例，以下语句将对 MAT 文件进行编译连接：

mex –f <MATLAB>\bin\msvc50engmatopts.bat *filename*.c

文件 watengmatopts.bat、wat11engmatopts.bat、 bccengmatopts.bat、msvc50engmatopts.bat 和 msvcengmatopts.bat 是 windows 下单机 C 语言 MAT 程序编译器组态文件，用户可以通过查看其中某个文件获得具体信息以便为自己的 C 程序进行编译连接。

在编写 MAT 文件代码时，MATLAB 为用户提供了一个 MAT 文件接口库。该库包含一个 MAT 文件读写子程序集，用户可以在自己的 C 或 Fortran 程序中调用该库中的函数。最好不要自己编写 MAT 的读写操作函数以防将来 MAT 文件格式可能发生变化。MAT 库函数都是以 mat 为前缀的，表 2-1 中列出了一些常用的 MAT 函数及其功能，用户可以通过调用这些函数实现 MAT 文件的操作。

表 2-1　常用 MAT 函数及其功能

MAT 函数	功　能
matOpen	打开一个 MAT 文件
matClose	关闭一个 MAT 文件
matGetDir	从一个 MAT 文件中获取 MATLAB 数组列表
matGetFp	将一个 ANSI C 文件指针指向一个 MAT 文件
matGetArray	从一个 MAT 文件中读取一个 MATLAB 数组
matPutArray	向一个 MAT 文件中写入一个 MATLAB 数组
matGetNextArray	从一个 MAT 文件读取下一个 MATLAB 数组
matDeleteArray	从一个 MAT 文件中删除一个 MATLAB 数组
matPutArrayAsGlobal	将一个 MATLAB 数组放在一个 MAT 文件中，使 load 命令能够将该数组放入 MATLAB 工作平台中

2.3.4　MATLAB 引擎的使用

MATLAB 引擎的使用主要是通过 MATLAB 引擎库来实现的。MATLAB 引擎库中的子程序均以 eng 为前缀，表 2-2 中列出了 C 语言 MATLAB 引擎库包含的子程序及其功能。

表2-2 C语言MATLAB引擎库子程序及其功能

子程序名	功 能
engOpen	开始一个MATLAB引擎
engClose	关闭一个MATLAB引擎
engPutArray	发送一个MATLAB数组到MATLAB引擎中
engEvalString	执行一个MATLAB命令
engOutputBuffer	创建一个缓冲区来存储MATLAB文本输出
EngGetArray	从MATLAB引擎中获取一个MATLAB数组

在Windows系统下使用MATLAB引擎采用的是ActiveX技术。MATLAB支持两种ActiveX技术：ActiveX控件容器和ActiveX自动化。ActiveX控件是集成于一个ActiveX控件容器（例如MATLAB图形窗口）中的可视化可编程应用程序组件，这种ActiveX使用比较简单，这里不作介绍。

ActiveX自动化技术使得MATLAB能够控制或被控于其他ActiveX组件。当MATLAB被控于某个组件时，MATLAB被称为自动化服务器，相反情况下则被称为自动化用户。MATLAB作为自动化服务器时能够在MATLAB工作平台中执行命令或从工作平台中输入输出矩阵，而作为自动化用户时可以通过使用M文件对自动化服务器进行演示和操作。前一种方式是MATLAB较为常见的工作方式。

MATLAB作为ActiveX用户时支持以下操作：

（1）在MATLAB中通过创建一个MATLAB ActiveX类实例来创建ActiveX控件和服务器，每一个实例代表一个对象接口。MATLAB有两种实例创建函数：

actxcontrol：创建一个ActiveX控件

actxserver：创建一个ActiveX自动化服务器

（2）接口操作：以上两个创建函数都返回一个MATLAB ActiveX对象，代表所创建对象的缺省接口。对该接口进行操作的函数主要有以下几种：

set: 设置接口属性

get: 获取当前接口的属性值

invoke: 在接口中调用一个MATLAB自动化方法

propedit: 请求接口显示其内置的属性页

release: 释放一个activex对象

delete: 删除一个activex对象

MATLAB作为ActiveX自动化服务器时，用户程序可以通过自动化技术实现MATLAB命令执行并从MATLAB工作平台中获取mxArray数据。使用MATLAB作为自动化服务器通常采用以下两个步骤实现：

（1）首先用户必须明确所使用的控件调用ActiveX自动化服务器的方法。MATLAB作为ActiveX对象在注册表中的名称为MATLAB.Application。虽然用户使用的控件调用MATLAB自动化服务器的方法可能不同，但所有控件都需要这个名称来标示服务器。

（2）调用MATLAB自动化执行方法执行用户控件所需的操作。MATLAB自动化方法所使用的数据类型是ActiveX自动化数据类型，这种数据类型与具体的编程语言无关，是根据ActiveX自动化协议指定的。所有ActiveX控件都应该能够识别并对这些数据类型进行操作。MATLAB自动化方法主要有以下三种：

- STR Execute(BSTR Command)：该命令接受单个的字符串 command，该字符串可以包含任何能够在 MATLAB 提示符后键入的命令。MATLAB 执行根据 command 命令并返回字符串或显示图形；
- void GetFullMatrix(
                BSTR Name,
                [in] BSTR Workspace,
                [in, out] SAFEARRAY(double)* pr,
                [in, out] SAFEARRAY(double)* pi)：

  该命令从指定的工作平台中返回一个空的一维或二维实数或复数 mxArray。
- void PutFullMatrix(
                [in] BSTR Name,
                [in] BSTR Workspace,
                [in] SAFEARRAY(double) pr,
                [in] SAFEARRAY(double) pi)：

  该方法将一个空的一维或二维实数或复数 mxArray 交给指定的工作平台。

以下是一个 VB 作为自动化用户调用 MATLAB 自动化方法的例子：

```
    Dim MATLAB As Object
    Dim Result As String
    Dim MReal(1, 3) As Double
    Dim MImag() As Double
    Dim RealValue As Double
    Dim i, j As Integer

    Result = MATLAB.Execute("a = [1 2 3 4; 5 6 7 8;]")
    Call MATLAB.GetFullMatrix("a", "base", MReal, MImag)
    For i = 0 To 1
    For j = 0 To 3
         RealValue = MReal(i, j)
         Next j
Next i
```

2.4 实例讲解

【实例一】：通过 MATLAB 的字符矩阵运算获取 MATLAB 当前使用的字体信息。

分析：解决这个问题的根本依据是：字符数组在磁盘中按照该字符的 ASCII 码对应整数来存储，因此通过将数值转换为字符可以了解用户计算机所使用的不同字体。基本 ASCII 码集中可打印的字符是由整数 32 到 127 表示（32 以下代表不可打印字符）的，128 以上的

整数表示扩展 ASCII 码字符集，当系统选择不同的字体时，这些整数将被解释为何种字符会有所不同。因此，通过将字符矩阵转换为数值矩阵可以将不同的字符与相应的整数相对应，然后观察这些整数在不同字体下所对应的字符有何变化。

解决方法：

步骤一：首先将部分 ASCII 码集中可打印的字符与相应的整数对应起来。用以下语句将这些整数排列在一个 2×16 的数组之中：

```
F =reshape(32:63,2,16)                                   (1)
```

步骤二：观察基本 ASCII 码集中可打印的部分字符（分别由整数 32 到 63 表示）。在命令窗口下键入语句：

```
char(F)
```

显示结果如下：

```
ans =
    "$&(*,.02468:<>
     !#%')+-/13579;=?
```

步骤三：使用整数 F+128 来表示扩展 ASCII 码字符集中的可打印字符。输入命令：

```
char(F+128)
```

用户将看到当前字体下整数 160 至 191 所对应的字符。

步骤四：观察不同字体下字符与整数的不同对应。改变命令窗口所使用的字体，用户将看到语句（1）的显示结果发生的变化。

【实例二】：使用 MATLAB 的功能函数。

分析：为了演示功能函数的使用方法，使用下述的实例：

MATLAB 有这样一个演示函数 humps：

```
function y=humps(x)
    y=1./((x-.3).2+.01)+1./((x-.9).^2+.04)-6;
```

对一个[0,1] 区间上间隔为 0.002 的点集进行 humps 运算并绘图：

```
x=0:.002:1;
y=humps(x);
plot(x,y)
```

可得如图 2-2 所示的曲线。

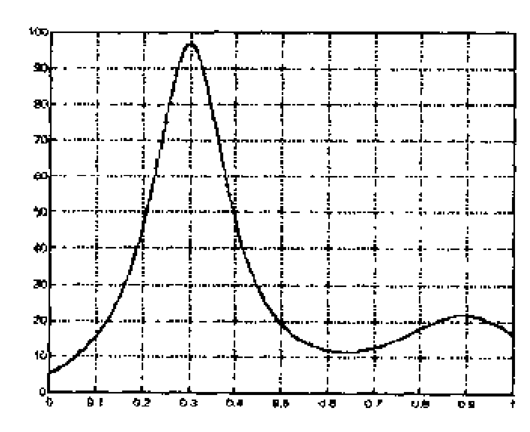


图 2-2 humps（x）曲线

使用功能函数对函数 humps 求取局部最小值以及曲线下方的面积。例外，从图中可以看出，该曲线在[0,1]区间内始终不为零，使用功能函数对此进行验证。

解决方法：

步骤一：使用功能函数 fminsearch 计算函数 humps 曲线的极小值。Fminsearch 函数有两个参数，第一个参数表示 humps 函数的句柄，第二个参数表示对极小值的粗略估计。输入语句：

```
fminsearch(@humps,.5)
```

计算结果为：

```
ans=0.6370
```

与所绘制的曲线相符。

步骤二：使用功能函数 quadl 计算[0，1]区间曲线下方的面积。输入语句：

```
quadl(@humps,0,1)
```

计算结果为：

```
ans=29.8583
```

步骤三：使用求零函数 fzero 在区间[0，1]不取零值。输入语句：

```
fzero(@humps,.5)
```

计算结果为:

```
ans=-0.1316
```

验证函数 humps 在[0，1]区间确实不取零值。

【实例三】：timestwo.c 可以用来计算一个标量的 2 倍数，该文件的代码如下，怎样在 MATLAB 中调用这个文件进行计算？

```
#include <math.h>
void timestwo(double y[], double x[])
{
y[0] = 2.0*x[0];
return;
}
```

分析：在 MATLAB 中调用 C 程序是通过编写 MEX 文件来实现的。为此，需要将以上代码重新编写，以适应 MEX 文件的要求。由于编译器组态方法和 C 源代码文件的编译连接方法在第三节的内容中已经较为详细地介绍过，所以以下仅介绍怎样将这个程序改写为 MEX 文件。

解决方法：

步骤一：对 C 编译器进行组态。

步骤二：组态测试。

步骤三：编写 MEX 文件。代码如下：

```
#include "mex.h"
    void timestwo(double y[], double x[])
    {
    y[0] = 2.0*x[0];
    }
    void mexFunction( int nlhs, mxArray *plhs[],
    int nrhs, const mxArray *prhs[] )
    {
    double *x,*y;
    int mrows,ncols;
```

```
    if(nrhs!=1)
    {
        mexErrMsgTxt("One input required.");
    }
    else if(nlhs>1)
    {
    mexErrMsgTxt("Too many output arguments");
    }
        mrows = mxGetM(prhs[0]);
        ncols = mxGetN(prhs[0]);
        if( !mxIsDouble(prhs[0]) || mxIsComplex(prhs[0]) ||!(mrows==1 && ncols==1) )
        {
        mexErrMsgTxt("Input must be a noncomplex scalar double.");
    }
        plhs[0] = mxCreateDoubleMatrix(mrows,ncols, mxREAL);
        x = mxGetPr(prhs[0]);
        y = mxGetPr(plhs[0]);
        timestwo(y,x);
}
```

2.5 小　　结

通过本章的学习，读者能够掌握MATLAB的一些高级编程技巧，从而能够使用MATLAB完成一些较为高级的任务。

除了基本的数组和矩阵，MATLAB 的一些其他数据类型也是非常有用的。在利用MATLAB 进行编程的过程中，用户可以利用 MATLAB 提供的各种数据类型来完成一些特殊的工作，例如观察系统字体情况等。另外，在本章的内容中还列举了大量实例对每个概念或计算方法进行详细介绍，用户可以通过理解这些实例来掌握这些内容以提高 MATLAB 的编程效率，例如使用矢量化和预分配方法。

实现应用程序间的交互对于每一种语言的高级程序员来说都是必须掌握的内容。MATLAB 与其他程序进行交互的方法较多，例如 MEX 文件、MAT 文件和 MATLAB 引擎的使用等。事实上这部分内容是非常复杂的，本章仅对这些内容做了大致的介绍，有兴趣的用户可以参阅相关的参考书。

到目前为止用户已经具备了一定的 MATLAB 编程能力，这将为今后图形系统的学习打下坚实的基础。

第 3 章　MATLAB 图形初步

在掌握了 MATLAB 的基本操作和编程方法的基础上，从本章开始，读者将进入 MATLAB 图形系统的学习。显然，图形比单纯的数据具有更好的可读性，所包含的信息量也更多。本章将向读者初步介绍 MATLAB 图形系统的强大功能，包括 MATLAB 图形种类、基本绘图方法、图形编辑方法。除了使用高级绘图命令进行数据的绘制，本章还介绍了图形对象及其句柄的概念以及怎样通过使用句柄来修改图形的属性，从而使图形完全符合用户要求。另外，在本章中还大致介绍了图形用户界面的创建方法和步骤。

3.1　MATLAB 基本图形及编辑方法

3.1.1　MATLAB 图形系统组成

图形系统具有非常强大的功能，它不但包含许多高级绘图函数，能够实现二维和三维数据的可视化、图像处理、动画和描述性图形等功能，还包含一些允许用户将图形完全客户化的低级命令。另外，用户还可以在自己的应用程序中建立图形用户界面。

从图形类型的角度可以将 MATLAB 图形系统分为以下几部分：

- 基本图形：这部分图形功能主要是实现用户数据的可视化。用户常常会希望能通过图形来尽可能地获得更多自身数据所描述的信息，这就要求 MATLAB 具有绘制向量和矩阵的能力。基本图形系统主要是通过使用各种不同语法结构的绘图函数 plot 来实现的。另外，为了增强数据图形的可读性，还可以定制图形所在窗口的外观并给图形所在的坐标轴添加标签、图例、标题和文本注释等；
- 网格和曲面图形：与基本图形相比，网格和曲面图形具有更好的可读性和更大的信息量。基本图形大多是二维的数据描述图形，而网格和曲面图形是三维的数据描述图形。同时，通过对网格和曲面图形进行镜头操作或添加灯光、设置透明度及缩放效果，可以使图形具有更强的真实感；
- 图像：MATLAB 也可以将二维数组显示为图像。与基本图形不同的是，图像的二维数组单纯地表示图像的颜色或亮度，通过将数据映射到特定的调色板中来显示，而基本图形的二维数据是用户数据，是用户获得的或通过计算分析得来的有具体意义的数据，基本图形根据数据来计算或选择所需的图形颜色。图像的显示使用函数 image 以及 colormap 来完成；
- 图形句柄：实际上 MATLAB 在执行绘图函数时是通过使用不同的图形对象来实现图形创建功能的，这些图形对象包括直线、文本、曲面等等。MATLAB 创建一个图

形对象时总会给该对象指定一个独一无二的标示符，这个标示符就称为句柄。通过使用句柄，用户可以方便地访问句柄所指定的对象，通过修改对象的属性使图形完全符合用户要求；

- 图形用户界面：图形用户界面是使用 MATLAB 图形对象创建的用户界面，用户可以通过界面中的用户控件来实现用户与应用程序间的交互。用户对控件的操作通过控件的回调函数来表示和实现。MATLAB 为建立图形用户界面提供了一个开发环境（GUIDE），使图形用户界面的创建变得非常简单；
- 动画：MATLAB 为创建动画提供两种较为简单的方法：一种是在屏幕上不断地擦除和重画某个对象，在重画过程中实现对象的不断移动；另一种是通过保存一系列的图片，然后采用类似于电影制作的方法在后台连续放映这些图片。

3.1.2 绘图基本过程

构造一个用户描述图形的过程大致可以分为以下七个步骤，用户可以根据自己的需要选择相应的步骤。为了清楚地说明每一步骤所做的工作，以贝塞尔函数绘制为例进行介绍。

（1）数据准备：

```
x = 0:0.2:12;
y1 = bessel(1,x);
y2 = bessel(2,x);
y3 = bessel(3,x);
```

（2）选择窗口和图形在窗口中的位置：

```
figure(1)
subplot(2,1,1)
```

（3）调用绘图函数：

```
h = plot(x,y1,x,y2,x,y3);
```

（4）设置图形对象的属性，例如选择线型和标记符号：

```
set(h,'LineWidth',2,{'LineStyle'},{'--';':';'-.'})
set(h,{'Color'},{'r';'g';'b'})
```

（5）设置其他能够增强图形可视性的属性，例如设置坐标轴范围、比例尺和网格线：

```
axis([0 12 –0.5 1])
grid on
```

（6）标注图形以增强图形的可读性，例如使用坐标轴标签、图例和文本注释：

```
xlabel('Time')
ylabel('Amplitude')
legend(h,'First','Second','Third')
title('Bessel Functions')
[y,ix] = min(y1);
text(x(ix),y,'First Min \rightarrow','HorizontalAlignment','right')
```

（7）输出图形：

```
print -depsc -tiff -r200 myplot
```



图3-1给出了以上例程的绘制结果。

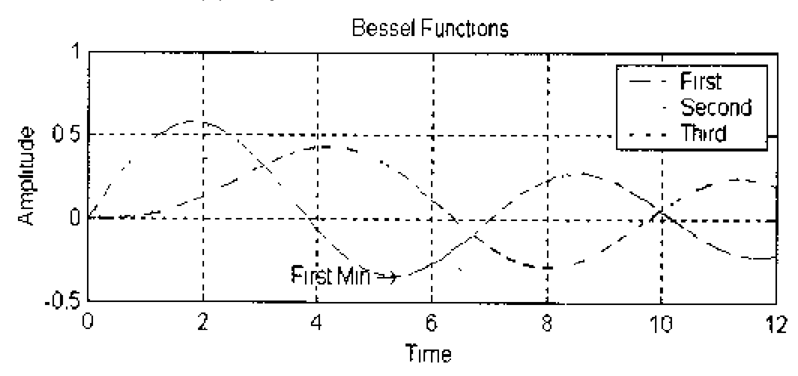


图 3-1　贝塞尔函数绘图结果

3.1.3 常用图形函数

MATLAB 图形系统包括许多图形绘制函数，大致可以分为以下几类：

- 二维图形绘制函数，例如 plot、loglog、semilogx 等；
- 三维图形绘制函数，例如 plot3、mesh、surf 等；
- 特殊图形绘制函数，例如 bar、contour、sphere 等；
- 图形对象创建函数，例如 figure、axes、line、text、patch 等；
- 图形对象句柄操作函数，例如 gca、set、get、findobj 等；
- 图形硬拷贝和打印函数，例如 print 等。

表 3-1 中列出了一些常用图形函数的名称和用途。

表 3-1　常用图形函数名称及功能

类　型	函 数 名	功　　能
二维图形	Plot	绘制线条图形
	Loglog	绘制对数坐标图形
	semilogx ,semilogy	绘制半对数坐标图形
	Plotyy	绘制双 y 轴图形
三维图形	plot3	在三维空间中绘制曲线和点
	Mesh	绘制三维网格曲面
	Surface	绘制三维彩色曲面
	fill3	绘制三维填充多边形
特殊图形	bar,bar3	绘制直方图
	Hist	绘制柱状图
	pie,pie3	绘制饼形图
	stem,stem3	绘制离散序列图
	contour,contour3	等高轮廓图
创建句柄	Figure	创建图形窗口
	Axes	创建坐标轴
	Line	创建线条
	Uicontrol	创建用户界面控件
	Uimenu	创建用户界面菜单
句柄操作	Set	设置对象属性
	Get	获取对象属性
	Findobj	根据指定属性值查询对象句柄

（续表）

类 型	函 数 名	功 能
句柄操作	Delete	删除对象
	Gco	获取当前对象句柄
	Gcbf	获取当前回调窗口句柄

3.1.4 图形编辑方法

为了增强图形的可读性，MATLAB 提供了许多缺省的有效方法，如坐标轴的设置、颜色的设置、线型的设置以及标记符等。然而，如果创建的是具有介绍性质的图形，用户希望能改变系统默认的设置，添加一些自身的描述性标记、标题、图例和注释，也就是要根据需要对图形进行编辑。

如果用户允许在 MATLAB 图形窗口中使用图形编辑模式，则可以使用点击编辑方式来编辑用户图形。在这种模式下，用户可以通过双击图形对象，修改该对象的属性值来改变对象在图形中的显示方式。这些行为是通过图形用户界面访问对象的属性值来实现的。图形编辑模式下的窗口如图 3-2 所示。

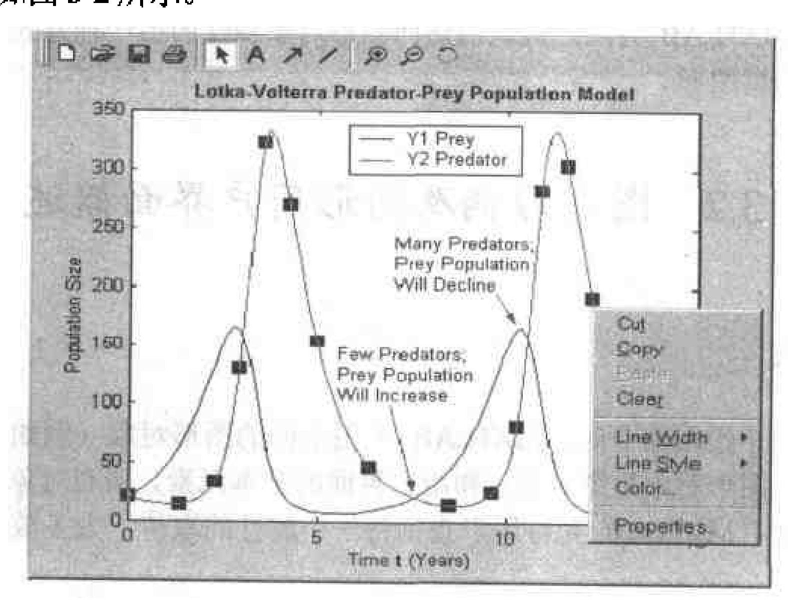


图 3-2 图形编辑模式窗口外观

在用户能够在图形窗口中编辑图形之前，用户必须激活图形编辑模式。启动图形编辑模式的方法有这样几种：在图形窗口的 Tools 菜单中选择 Edit Plot 选项；在图形窗口的工具栏中点击选择按钮（参见图 3-2）；在 Edit 或 Insert 菜单下选择一个选项，例如 Edit 菜单的 Axes Properties 选项；在 MATLAB 命令窗口下运行 plotedit 命令。

图形编辑分为两个步骤：首先利用鼠标选定编辑对象进行交互式图形编辑，然后利用 MATLAB 菜单栏或文件菜单中的功能进行编辑。

首先来选定编辑对象：将鼠标移动到需要的对象上并点击它。如果需要同时选择多个对象，则在点击对象的同时按下键盘上的 shift 键。用户可以同时对所选的所有对象进行操作，例如，如果希望删除一个文本注释和箭头注释，选择这两个对象，然后选择 Edit 菜单的 Cut 选项。如果希望取消选定的对象，将鼠标移动到别处点击鼠标左键，也可以按下 shift 键同

时对所选对象再次点击以取消选定。

基本的编辑任务包括：

- 在图形中选择图形对象；
- 剪切、拷贝、粘贴图形对象；
- 移动和重画图形对象；
- 编辑图形中的图形对象；
- 保存编辑结果；
- 改变坐标轴中的视图。

这些任务均可以通过菜单或按钮方便地完成。另外，如果用户选定了一个对象，那么就可以通过双击该对象或选择属性菜单来打开属性编辑器，通过修改对象的属性来进行图形编辑。

再次点击选择按钮或 Tools 菜单下的 Edit Plot 选项将退出图形编辑模式。

另外，如果用户希望在 MATLAB 命令行中进行图形编辑工作，或者用户正在创建一个 M 文件，那么还可以使用 MATLAB 函数来编辑用户创建的图形。充分利用 MATLAB 图形句柄系统的优点，用户可以使用 set 或 get 函数根据句柄来修改图形中对象的属性。事实上，图形编辑模式是 MATLAB 为用户提供的访问图形对象属性的另一种方式。用户可以同时使用这两种方式来编辑图形。

3.2　图形句柄及图形用户界面概述

3.2.1　图形对象

当用户使用一个绘图命令时，MATLAB 使用不同的图形对象（例如线条、文本和曲面等）来创建图形。图形对象是图形显示和用户界面的基本元素。所有对象都具有控制本对象行为和外观的属性。MATLAB 允许用户查询每一个属性的取值，大多数属性都可以进行设置。

MATLAB 创建一个图形对象时总要给这个对象指定一个独一无二的标示符，这个标示符称为图形对象的句柄。用户可以使用这个句柄来访问对象的属性。在以下两种情况下图形句柄是非常有用的：

- 修改图形的外观；
- 通过编写直接对图形对象进行创建和操作的 M 文件来实现符合用户要求的绘图命令。

显然，句柄的使用是与图形对象的类型密切相关的，表 3-2 列出了 MATLAB 系统所有图形对象的种类。

图形对象间是有一定相互依赖关系的，例如，线条对象需要使用坐标轴对象作为参考框架，而坐标轴对象又只能存在于一个图形窗口中。图形对象继承关系表就是反映图形对象间依赖关系的树形结构层次列表。图 3-3 列出了继承关系表的结构。

表 3-2 MATLAB 系统图形对象种类表

对 象 类 型	对 象 描 述
根对象（root）	相当于计算机屏幕
图形窗口对象（figure）	显示图形和用户界面的窗口
轴（axes）	在图形窗口中显示图形的坐标轴
用户控件（uicontrol）	执行用户交互响应函数的用户界面控件
用户菜单（uimenu）	用户定义图形窗口的菜单
用户文本敏感菜单（uicontextmenu）	右击图形对象调用的弹出式菜单
图像（image）	基于像素的二维图片
灯光（light）	影响面片和曲面对象的光源
线条（line）	函数 plot、plot3 等使用的线条
面片（patch）	有边界的填充多边形
矩形（rectangle）	从椭圆到矩形变化的二维形状
曲面（surface）	将数据作为 x-y 平面高度创建的三维矩阵数据描述
文本（text）	字符串

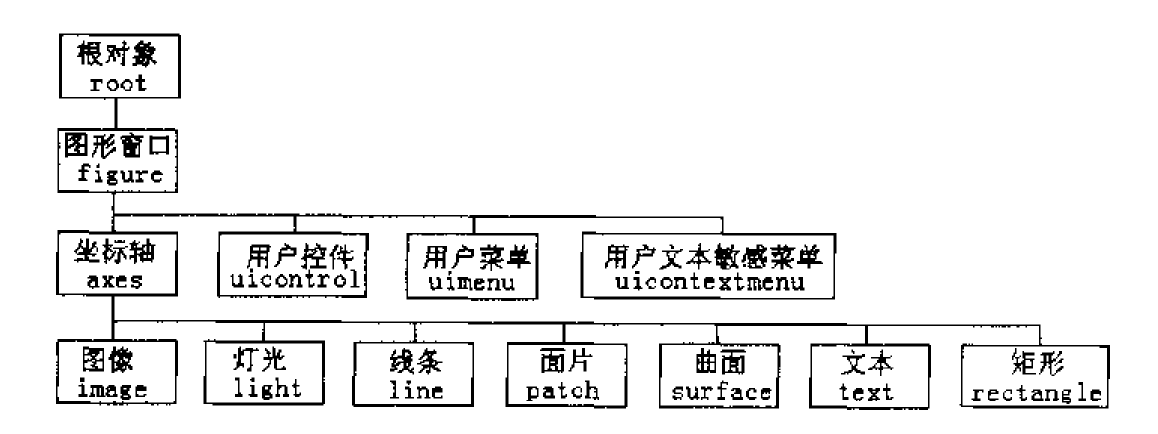


图 3-3 图形对象继承关系表

每一种类型的图形对象都有一个相应的创建函数，这个创建函数使用户能够创建这一类对象的一个实例。对象创建函数与该函数创建的对象名相同，例如，函数 text 将创建一个文本对象，figure 函数将创建一个图形窗口对象。MATLAB 高级绘图函数（例如 plot 和 surf）将调用这些低级函数来绘制各自的图形。

除了在表 3-1 列出的对象操作函数 findobj、delete、gco、gcbf、set、get 以外，还有以下几个函数专用于图形对象的操作：

copyobj：拷贝图形对象；

gca：返回当前坐标轴的句柄；

gcf：返回当前图形窗口的句柄；

gco：返回当前对象的句柄。

3.2.2 图形对象属性

图形对象的属性可以控制对象外观和行为许多方面的性质。对象的属性既包括对象的一般信息，例如对象类型、父类和子类、是否可见等，又包括特殊类型对象独一无二的信息。

例如，从任意给出的 figure 对象中用户可以获得这样一些信息：窗口中最后按下的键、指针的位置以及最近一次选择的菜单项等等。MATLAB 将图形对象的信息组织成一个层次表，并将这些信息储存在该对象的属性中。例如，root 属性表包括当前图形窗口的句柄和当前的指针位置，figure 属性包括其子对象的列表并始终跟踪窗口中发生的特定事件，axes 属性则包含有关其子对象使用图形窗口映射表的方法以及 plot 函数使用的颜色命令。

一些属性对于所有的图形对象来说具有相同的含义。表 3-3 列出了这些共同属性：

表 3-3　图形对象的共同属性

属 性 名 称	包 含 信 息
BusyAction	控制 MATLAB 处理特定对象回调函数中断的方法。
ButtonDownFcn	定义当按钮按下时执行的回调函数
Children	所有子对象的句柄
Clipping	剪贴板允许或不允许模式（实际上仅对 axes 的子对象有意义）
CreateFcn	创建本类对象时执行的回调函数
DeleteFcn	用户发出销毁本对象命令时调用的回调函数
HandleVisibility	控制对象句柄的获得方式（命令行中获得和会调函数中获得）
Interruptible	决定回调函数是否能够被随后调用的回调函数中断
Parent	对象的父对象
Selected	表明对象是否被选择
SelectionHighlight	定义对象是否使用可见方式表明被选择状态
Tag	用户定义的标签
Type	对象类型（line、figure、text 等）
UserData	用户希望与对象联系起来的任意数据
Visible	决定对象是否可见

所有对象属性都有缺省值。用户可以修改属性的设置使图形符合用户自己的要求。有两种方法可以用来改变对象的属性，一种是在创建对象时设置，另一种是在创建完成后进行设置。

创建对象时指定该对象的属性值：用户可以在调用对象创建函数或高级绘图函数（例如 plot 和 surf）时将指定的属性值作为该函数的参数，从而设置对象的属性。例如：

```
plot(x,y,'LineWidth',1.5)
```

将使用由 LineWidth 属性指定的线宽为 1.5 个点（一个点等于 1/72 英寸）的线条对数据 x 和 y 进行绘制。

指定一个已存在对象的属性值：用户可以使用 set 命令来修改一个已存在对象的属性值。许多绘图函数会返回被创建对象的句柄，所以用户可以在调用绘图之后使用 set 命令对对象进行修改。例如，以下语句将绘制一个 5×5 的矩阵（根据每一列绘制一个曲线，共 5 条曲线），然后将设置曲线对应对象的 Marker 属性为 s(正方形)，MarkerFaceColor 为 g（绿色），绘图结果如图 3-4 所示。

```
h = plot(magic(5));
set(h,'Marker','s',MarkerFaceColor','g')
```

以上语句中的 h 是一个包括 5 个元素的向量，每一个元素都是图形中一个线条的句柄。set 将所有线条对象的 Marker 属性和 MarkerFaceColor 属性设为相同的数值。如果用户希望给每个线条设置不同的属性值，用户可以通过使用单元数组来存储这些需要的属性值，然后

将该单元数组传递给 set 命令。

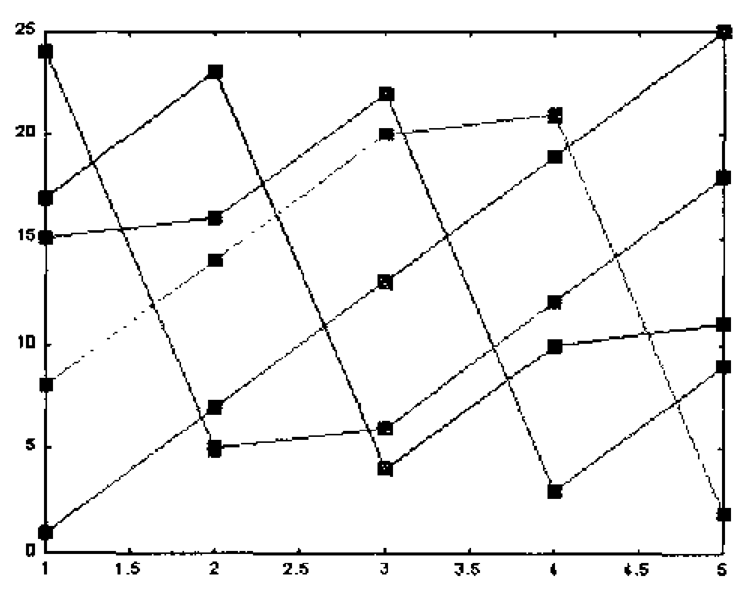


图 3-4 标记为绿色正方形的 magic 函数图形

对于已经存在的图形对象，也可以通过查找句柄的方法设置图形对象的属性。使用 findobj 函数可以使用户找到与指定属性值相同的对象的句柄，通过属性值或多个属性值的联合，用户可以很快从众多的图形对象中找到指定的对象。例如，假设用户希望找到具有正方形蓝色表面标记的线条。以下语句将获得所有线条对象的句柄：

```
h = findobj('Type','line');
```

为了找到具有正方形蓝色标记的线条，使用以下语句：

```
h = findobj('Type','line','Color','b', 'Marker','s' );
```

如果用户还知道一些其他的信息，例如对象所在的图形窗口和坐标轴，用户可以使用这些信息定义对象搜索的起始点，从而提高搜索效率。例如，以下语句将仅在当前坐标轴中寻找具有正方形蓝色标记的线条：

```
h = findobj(gca,'Type','line','Color','b', 'Marker','s' );
```

既然 findobj 函数返回的是搜索到的句柄，那么用户就可以使用该函数的输出作为 set 函数的参数，从而修改已存在对象的属性。例如，以下语句将搜索所有红色的线条并将其线型改为点线：

```
set(findobj('Type','line','Color','red'),'LineStyle',':')
```

3.2.3 图形用户界面

图形用户界面（GUI）是使用图形对象（例如按钮、文本框、滚动条和菜单等）创建的用户界面。通常这些对象对计算机用户而言都有明确的含义，例如移动滚动条将会改变数值，按下 OK 按钮将完成并应用用户的设置，同时设置对话框消失。当然用户必须保证这些不同对象间能够协调地工作。MATLAB 用一个包含多种不同风格用户控件对象的图形窗口代表用户界面。用户必须对每一个对象进行编程，使用户在 GUI 中的行为能够达到相应的目的。

实现一个 GUI 的过程包括两个基本任务：一是 GUI 的组件布局，另一个是 GUI 组件编

程。另外，用户还必须能够保存并发布自己的 GUI，使得用户开发的图形界面能够真正得到应用。所有这些功能都能通过图形用户界面开发环境 GUIDE 来完成。GUIDE 首先是一个组件布局工具集，能够生成用户所需的组件资源并保存在一个 FIG 文件中；其次，GUIDE 还将生成一个包含 GUI 初始化和发布控制代码的 M 文件，该文件为回调函数（用户在图形界面中激活某一控件时要执行的函数）提供了一个框架。事实上，用户也可以通过编写调用组件函数的 M 文件来实现 GUI 中所有组件的布局，但是使用 GUIDE 交互式的组件布局功能将会大大减小工作量。GUIDE 可以为 GUI 同时生成以下两个文件：

一个 FIG 文件：该文件包括 GUI 的图形窗口和所有子对象（包括用户控件和坐标轴）的完全描述以及所有对象的属性值。

一个 M 文件：该文件包括用户用来发布和控制界面和回调函数（这里作为子函数）的各种函数。该文件中不包含任何组件的布置信息。图 3-5 显示了一个 GUI 的部分结构。

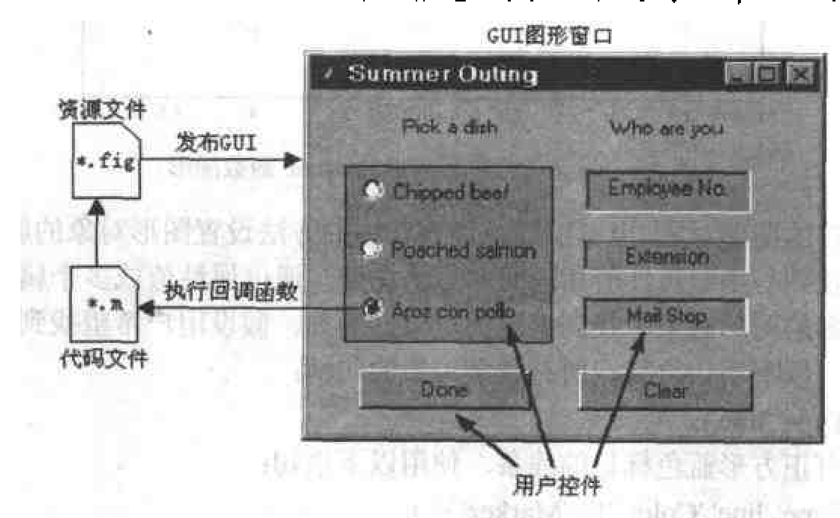


图 3-5　GUI 部分结构

3.2.4　创建 GUI 过程

MATLAB 提供以下几种组件布局工具：

- 组件布局编辑器：添加和安排图形窗口中的对象；
- 排列工具：排列对象的相对次序；
- 属性编辑器：检查和设置属性值；
- 对象浏览器：观察本次运行中图形对象句柄的层次关系；
- 菜单编辑器：创建图形窗口菜单和文本敏感菜单。

这些工具都集成在布局编辑器界面中，使用 guide 命令将显示该界面。如果要对一个已存在的 GUI 进行布局，可以使用菜单打开该 GUI 或使用以下命令装载已存在的图形窗口（mygui）：

```
guide mygui.fig
```

在添加需要布置的组件之前，应该使用 GUIDE 应用程序选项对话框对 GUI 进行组态。通过组件布置编辑器 Tools 菜单的 Application Options 选项来打开 GUIDE 应用程序选项对话框。在该对话框中，用户可以决定是否需要 GUIDE 为 GUI 生成 M 文件以及其他选项。

组态完成后就可以使用组件布局工具来布置用户所需的组件。通过使用布局工具，用户可以添加所需的用户控件对象并设置所需的属性。布局完成并存盘后，所有的对象信息就保存在相应的FIG文件中了。

下一步要对 GUIDE 生成的或用户自己编写的 M 文件进行编程来实现用户界面的交互功能。编程工作分为以下几个部分：

- 理解M文件：如果GUI的M文件是由GUIDE创建的，那么用户需要理解GUIDE创建的函数的意义，从而进行下一步编程；
- 管理GUI数据：MATLAB 提供一个句柄结构体来方便地访问GUI中的所有组件句柄，用户还可以使用这个结构体来存储M文件所需的全局数据；
- 设计交叉平台的兼容性：GUIDE 提供一个设置方法来保证用户 GUI 在不同平台上的良好外观。
- 回调函数编程与应用：用户对象的回调函数中有一些回调函数属性，用户可以通过设置这些属性来获得所需的操作；
- GUI图形窗口行为控制。

以上这些内容将在以后的学习过程中向读者详细介绍。

3.3 动　　画

3.3.1 MATLAB 动画图形方法介绍

MATLAB 提供两种产生动画图形的方法：

- 在屏幕上连续不断地擦除并重画对象，在每一次重画过程中体现对象的微小变化，从而形成动画效果；
- 事先存储一系列的图片，再使用类似于电影放映的方法在后台连续显示这些图片。

第一种方法对简单物体在固定场景中移动最为有效。该方法通过简单的绘制、擦除、再绘制实现动画效果，着色迅速准确，对快速移动的小物体比较适合。

第二种方法更适合于动画每一帧都比较复杂，不易快速重画的情况。由于动画的每一帧都是提前创建的，因此在回放过程中初始的绘图时间是不重要的。这种方式的动画并不在实时放映时为每一帧着色，只是简单地回放预先着好色的帧。

3.3.2 擦除模式方法

当动画的帧与帧之间的差异很小时，使用 EraseMode 属性对于创建简单图形的长序列动画来说是非常适合的。这里通过分析一个布朗运动（分子运动）模拟过程实例来说明该属性的使用方法。

首先定义待模拟的分子的个数。为了尽量减小帧与帧之间的差距，分子个数不宜过多。这里取20个：

```
n = 20
```

然后定义一个温度或分子运动速率，例如：

```
s = .02
```

以上两个参数的取值依赖于用户计算机的运行速度。一般而言，计算机运行速度越快，分子个数可以选择得越大，而温度或分子运动速率可以选择相对小一些。下面在给定的坐标范围内（-1/2≤x≤1/2，-1/2≤y≤1/2）创建 n 个随机点：

```
x = rand(n,1)-0.5;
y = rand(n,1)-0.5;
```

在边界为-1 和+1 的正方形区域内绘制这些随机点。保存随机点组成的向量的句柄，然后设置该向量的 EraseMode 属性值为 xor。这将通知 MATLAB 图形系统在点的坐标发生变化时不要对整个图形进行重画，而是使用一个异或操作来重新绘制该点邻域内的背景颜色。

```
h = plot(x,y,'.');
            axis([-1 1 -1 1])
            axis square
            grid off
        set(h,'EraseMode','xor','MarkerSize',18)
```

下面就可以开始动画过程了。使用一个通过 Ctrl+C 终止的 while 死循环，在每一次循环中给点的坐标添加一个小的正态分布的随机噪声，然后通过改变图形的 XData 和 YData 属性来实现分子的移动效果。

```
while 1
drawnow
x = x + s*randn(n,1);
y = y + s*randn(n,1);
set(h,'XData',x,'YData',y)
end
```

图 3-6 至图 3-8 给出了该动画过程的几帧图片。由动画过程可知，分子在不断地作扩散运动。用户可以通过运行该程序来估计一个点或所有点移动到正方形外面需要的时间。

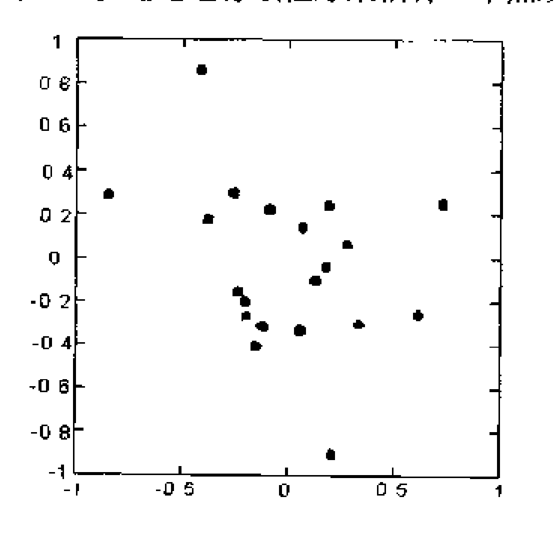


图 3-6 擦除模式动画起始帧

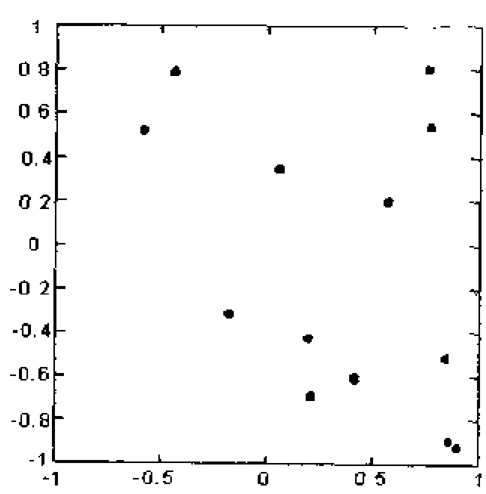


图 3-7 擦除模式动画中间帧 1

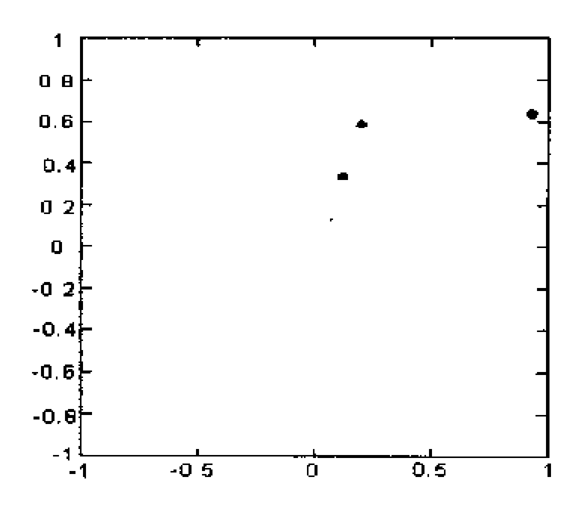


图 3-8 擦除模式动画中间帧 2

3.3.3 电影放映模式

如果用户增加上例中分子的个数，例如使 n=300，那么用以上方法创建的动画过程将变得不那么流畅，每一个重画过程需要很长时间。这时，有必要将一些预先得到的帧存储为位图，然后使用电影放映的方式在后台顺序显示这些位图。

首先，定义要存储的帧数，本例中取 50 帧：

```
nframes = 50;
```

然后除了将 EraseMode 设置为缺省的属性值（normal）外，使用与上例相同的方法来创建第一幅图形：

```
    x = rand(n,1)-0.5;
    y = rand(n,1)-0.5;
    h = plot(x,y,'.');
    set(h,'MarkerSize',18);
    axis([-1 1 -1 1])
    axis square
grid off
```

下面开始移动分子并捕获生成的每一帧：

```
    for k = 1:nframes
    x = x + s*randn(n,1);
    y = y + s*randn(n,1);
    set(h,'XData',x,'YData',y)
    M(:,k) = getframe;
end
```

最后连续放映捕获的帧：

```
movie(M,30)
```

图 3-9 和 3-10 是这种放映方式下产生的动画帧。从该程序的运行过程可以看出，前 50


帧采用擦除方法得到的动画速度明显比之后的电影放映方式动画要慢许多。

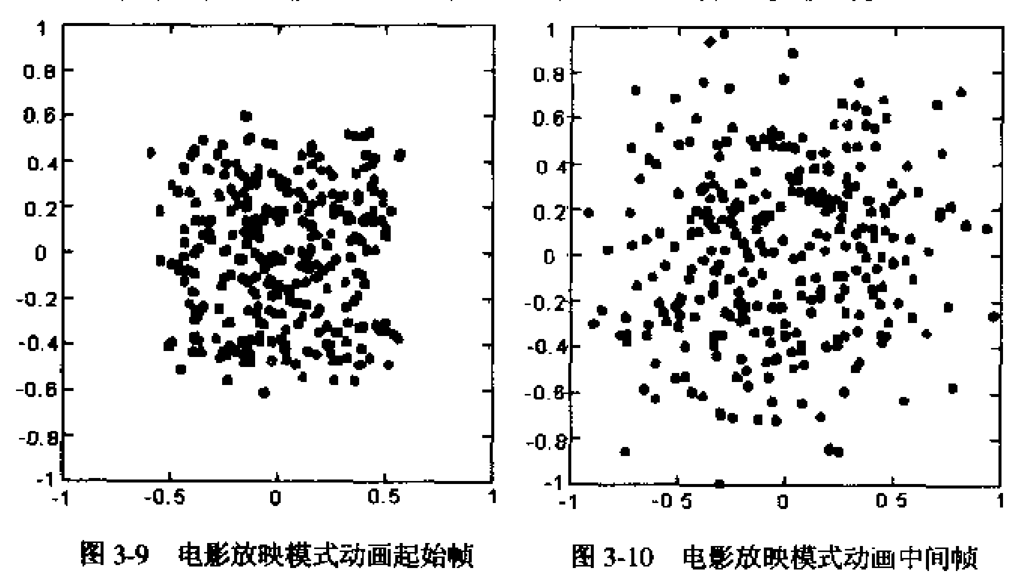


图 3-9　电影放映模式动画起始帧　　　　图 3-10　电影放映模式动画中间帧

3.4　实 例 讲 解

【实例一】：给图 3-4 所示的图形中的每个线条添加不同的标记，使用与线条相同的颜色给标记表面着色。

分析：为了给不同的对象设置不同的属性，用户需要定义两个单元数组：一个包含属性名，另一个包含所需的属性值。将两个单元数组作为 set 函数的输入，从而实现同时对所有所需对象进行属性设置。

解决方法：

步骤一：创建图形：

```
plot(magic(5));
```

步骤二：创建包含属性名的单元数组，prop_name 包括两个元素：

```
prop_name(1) = {'Marker'};
prop_name(2) = {'MarkerFaceColor'};
```

步骤三：创建包含属性值的单元数组 prop_values。prop_values 包含 10 个数值：5 个对应于 Marker，另外 5 个对应于 MarkerFaceColor。prop_values 是一个二维的单元数组，第一维表示数据将要应用于 h 中的哪个对象句柄，第二维表示数值将赋给哪个属性值。MarkerFaceColor 属性值总是设置为相应的线条颜色，线条颜色是通过使用 get 函数获取线条对象的 Color 属性得来的。

```
prop_values(1,1) = {'s'};
prop_values(1,2) = {get(h(1),'Color')};
prop_values(2,1) = {'d'};
prop_values(2,2) = {get(h(2),'Color')};
prop_values(3,1) = {'o'};
```

```
prop_values(3,2) = {get(h(3),'Color')};
prop_values(4,1) = {'p'};
prop_values(4,2) = {get(h(4),'Color')};
prop_values(5,1) = {'h'};
prop_values(5,2) = {get(h(5),'Color')};
```

步骤四：调用set函数来指定新的属性值。

```
set(h,prop_name,prop_values)
```

绘图结果如图3-11所示。

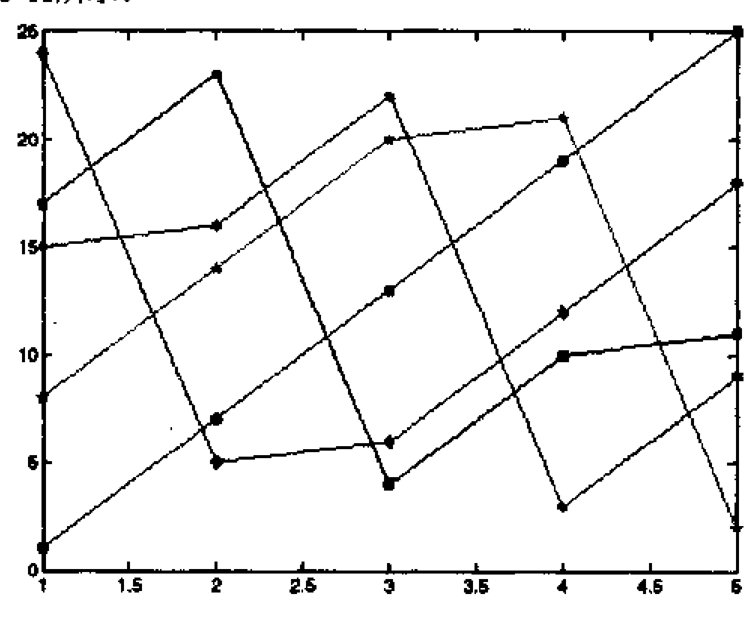


图 3-11 设置线条对象属性后的绘图结果

【实例二】：如何绘制二维函数 sinc 的三维网格图形。函数 sinc 的表达式为：sinc=sin(r)/r 式中 r 是当前点到原点的距离。

分析：为了显示一个双变量 x 和 y 的函数，首先要在变量 x 和 y 各自的函数域中创建由重复的行和列组成的矩阵 X 和 Y，使用 X 和 Y 来进行函数求值和绘图。函数 meshgrid 将由单个或两个向量 x 和 y 定义的域转换为矩阵 X 和 Y，然后在双变量求值函数中使用 X 和 Y 进行计算。X 的行向量是向量 x 的拷贝，Y 的列向量是向量 y 的拷贝。

解决方法：为了避免发生 0 作除数的情况（原点处），使用一个 MATLAB 函数 eps 来获得系统中最小的浮点数以保证 r 始终大于 0。

```
[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R)./R;
mesh(X,Y,Z,'EdgeColor','black')
```

此时的绘制结果如图 3-12 所示。

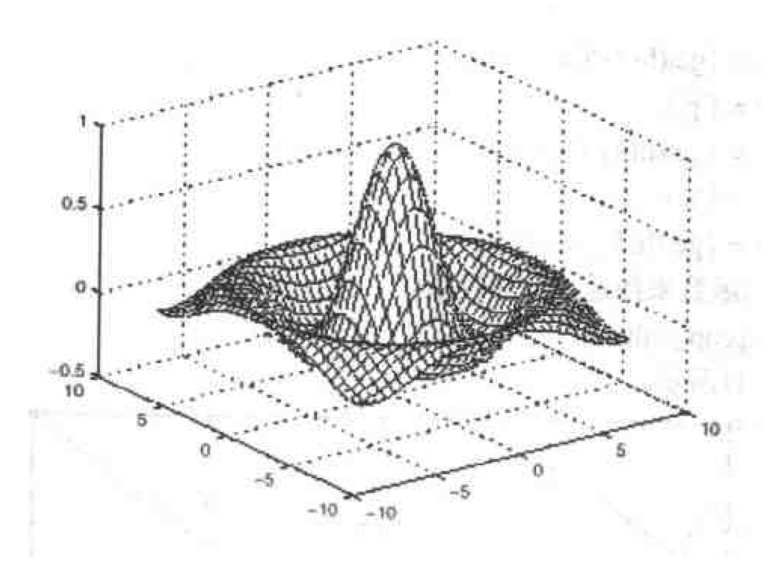


图 3-12　sinc 函数的网格图形

【实例三】：绘制上例中 sinc 函数的三维曲面图形，并尽量提高图形的可读性。

分析：曲面图形与网格图形基本类似，只不过曲面对其表面的矩形面也进行着色。矩形面的颜色由数据的函数计算结果（或 z 坐标值）和调色板决定。为了给图形增加可读性和真实感，可以给绘制的曲面增加灯光效果，同时使用 view 命令改变视点，使用户能够从空间的另一个点来更好地观察曲面。

解决方法：

步骤一：选择一个调色板来绘制函数的曲面图形，同时添加一个颜色条来说明数据到颜色的映射关系。

```
surf(X,Y,Z)
colormap hsv
colorbar
```

绘图结果如图 3-13 所示。

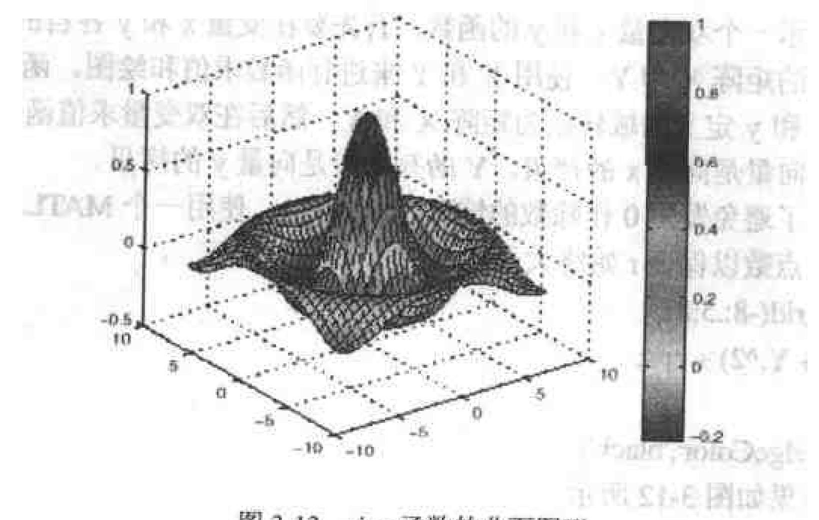


图 3-13　sinc 函数的曲面图形

步骤二：将曲面绘制成红色并删除网格线。

```
surf(X,Y,Z,'FaceColor','red','EdgeColor','none');
```

步骤三：添加效果。将一个灯光对象添加到镜头的左边并设置照明方式为 phone。

```
camlight left; lighting phong
```

步骤四：使用 view 命令改变视点为：高度角 65°、方位角-15°。

```
view(-15,65)
```

最终用户看到的曲面图形如图 3-14 所示，该曲面具有较好的真实感和可读性。。

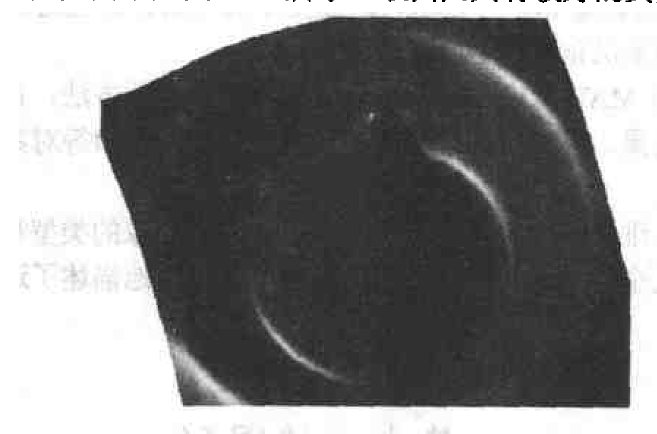


图 3-14　增加灯光效果的 sinc 函数曲面

3.5　小　　结

本章的主要目的是使读者初步了解和掌握 MATLAB 图形系统的各种功能。在本章中，读者可以学会如何创建 MATLAB 基本图形和进行图形编辑。在此基础上介绍了 MATLAB 图形系统较为底层的内容，包括图形对象及其属性。另外，本章中还大致介绍了图形用户界面（GUI）的概念和创建方法。

本章采用一些典型的实例来说明各种图形的创建和操作，通过理解这些例子，读者可以学会如何绘制简单的二维和三维图形，了解图形对象的概念，初步掌握句柄的使用和对象属性的设置方法，为今后深入研究 MATLAB 绘图打下良好的基础。在以后的学习中，读者应该将重点放在对图形的精确描述方面，力图掌握 MATLAB 图形系统各种绘图方式的每一个细节。

第 4 章　MATLAB 二维图形

MATLAB 提供许多将向量和矩阵显示为曲线图形的函数。通过对数据进行二维曲线绘制，可以从数据中获得更多的信息。

本章主要介绍了绘制 MATLAB 向量和矩阵二维图形的具体方法，详细描述了各种图形函数的使用方法及绘制结果，并阐明了如何通过设置坐标轴、窗口等对象的属性来定制图形的输出。

图像是一种特殊的二维图形，本章中介绍了 MATLAB 图像的类型和显示方法。最后，针对数据特性，介绍了几个特殊的数据描述绘图函数，并详尽地描述了这些函数的使用方法及其显示效果。

4.1　基本二维图形

4.1.1　二维图形创建

MATLAB 提供许多将数据显示为曲线图形的函数。表 3-1 中给出了几个常用的绘图命令，例如 plot、loglog、plotyy 等，这些函数都能够根据数据来绘制二维线条图形，它们之间的区别主要在于坐标轴刻画方式的不同。每一个函数在调用时都会根据数据自动选取相应的刻度范围来刻画坐标轴以适应输入参数向量或矩阵的要求。

根据输入参数形式的不同，plot 函数也会有不同的调用形式和绘图结果。

如果 y 为一个向量，那么 plot(y)将创建一个 y 的元素对应于其下标（即 x 轴代表向量 y 的元素的下标，y 轴代表向量 y 的元素值）的曲线图形。

如果指定两个向量参数 x，y，plot(x,y)将生成一个 y 对应于 x 的图形。例如，以下语句将创建一个 0 到 2π区间以步长π/100 变化的向量 x，然后计算向量 x 在该区间的正弦函数值。MATLAB 将向量 x 绘制在 x 轴上，将相应于向量 x 的正弦值绘制在 y 轴上。

```
x = 0 : pi/100 : 2*pi ;
y = sin (x);
plot ( x, y )
    grid on
```

plot 将根据输入参数自动选择相应的坐标轴范围和比例尺。以上语句的绘制结果如图 4-1 所示。

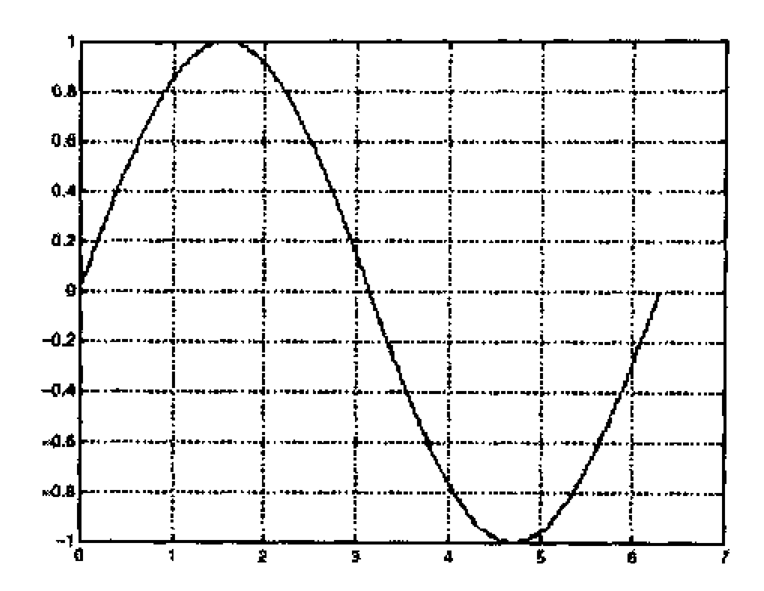


图 4-1　一个周期内的正弦函数曲线

可以通过使用成对的向量输入参数在 plot 函数的同一次调用中绘制多个图形，此时 MATLAB 将自动地循环使用预定义调色板中的颜色来区分不同的数据集。例如，如果在以上语句后再输入：

```
y2 = sin(t-0.25);
y3 = sin(t-0.5);
plot(t,y,t,y2,t,y3)
```

将会产生如图 4-2 所示具有三条曲线的图形。

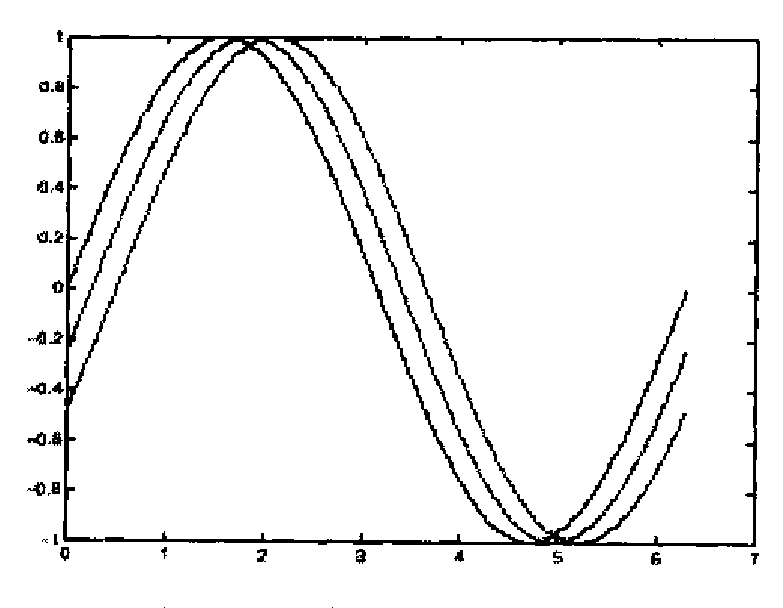


图 4-2　在同一个图形中绘制多条曲线

如果参数为一个单独的矩阵 Z 时，plot(Z)将为矩阵的每一列绘制一条曲线，x 轴将使用矩阵的行下标向量 1：m 来标注，其中 m 是矩阵的行数。例如：

```
Z = peaks;
```

将返回一个 49×49 的高斯分布矩阵。绘制这个矩阵：

```
plot(Z)
```

将产生如图 4-3 所示含有 49 条曲线的图形。

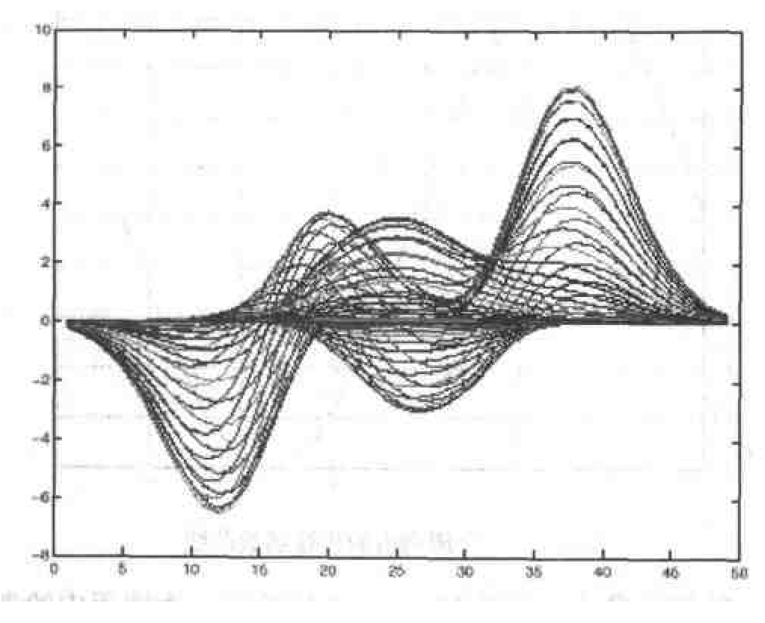


图 4-3　矩阵图形绘制

如果在调用 plot 时使用两个参数 x 和 y，其中至少有一个参数的行或列大于 1，那么将产生以下几种情况：

- 如果 y 是一个矩阵，x 是一个向量，则 plot(x,y)将产生 y 的行或列相对于 x 的多条曲线，每一条曲线使用设置好的不同线型或颜色。究竟使用 y 的行还是列取决于 x 的元素个数是符合 y 的行向量还是列向量。如果 y 是一个方阵，那么使用 y 的列。
- 如果 x 是个矩阵而 y 是个向量，那么 plot(x,y)将产生 x 的行或列相对于 y 的多条曲线。例如：

```
y = 1:length(peaks);
plot(peaks,y)
```

将产生如图 4-4 所示的结果。

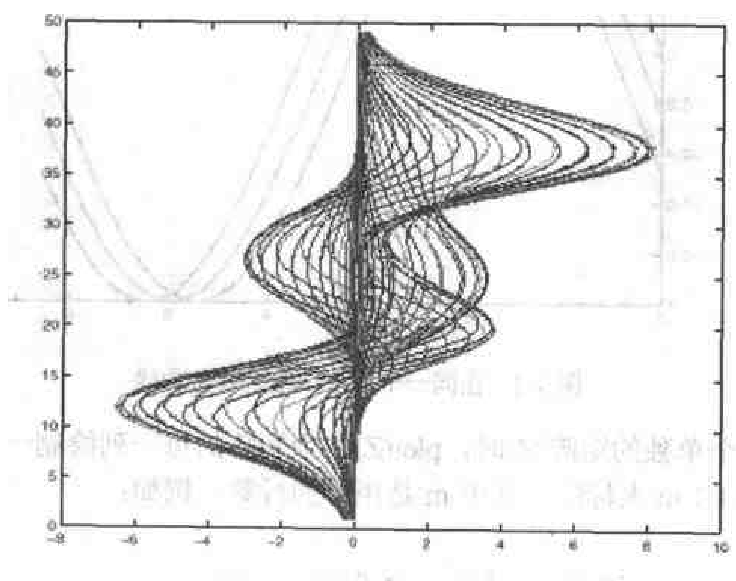


图 4-4　x 是矩阵 y 是向量时的绘图结果

● 如果 x 和 y 均是相同大小的矩阵，plot(x,y)将产生 x 的列相对于 y 的列的多条曲线。同样可以使用多个矩阵对作为 plot 的参数来生成多条曲线。不同的矩阵对之间可以有不同的矩阵维数。

当 plot 函数的参数是复数（虚部不为零）时，如果参数不止一个，plot 将会忽略参数的虚部。如果只有一个输入参数 Z，其中 Z 为一个复向量或复数矩阵，则 plot(Z)将绘制一个复数实部对应虚部的图形。这时 plot(Z)相当于：

```
plot ( real(Z),imag(Z))
```

例如，以下语句将绘制一个随机矩阵的特征值数据点：

```
plot(eig(randn(20,20)),'o','MarkerSize',6)
```

该随机矩阵的特征值矩阵由以下数据构成：

1.8023 + 3.7533i	1.8023 - 3.7533i	-0.8068 + 3.8624i	-0.8068 - 3.8624i
-3.5723 + 2.1144i	-3.5723 - 2.1144i	2.7566	-0.2786 + 2.6347i
-0.2786 - 2.6347i	-1.5926 + 2.5072i	-1.5926 - 2.5072i	1.5145 + 1.1639i
1.5145 - 1.1639i	1.2101	-3.1724	0.4843
-3.0848	-1.8623 + 0.2209i	-1.8623 - 0.2209i	-1.0366

绘制结果如图 4-5 所示。

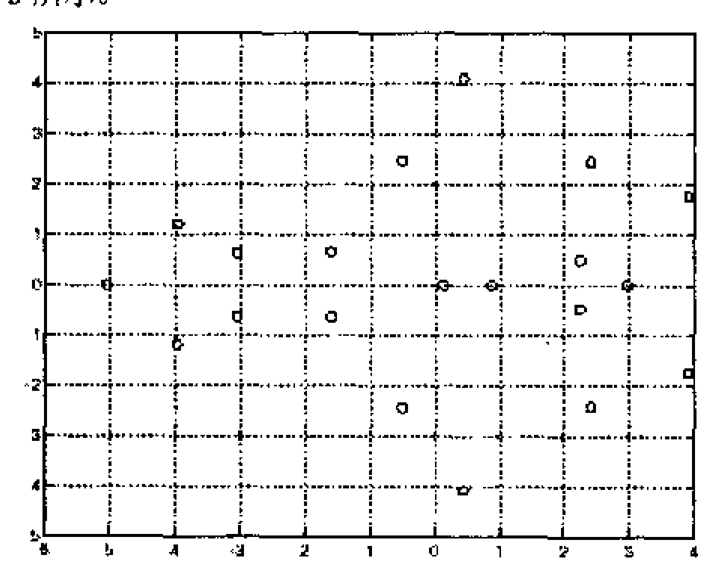


图 4-5 复数数据图形绘制

MATLAB 没有提供同时绘制多个复数矩阵的方法，如果需要同时绘制，必须将虚部和实部严格区分开，然后作为输入参数对调用 plot 实现。

plotyy 函数能够使用两个 y 轴（分别位于图形左边和右边）来绘制两个数据集。在该函数中，用户可以为每个数据集调用不同的绘图函数，例如，以下语句将同时绘制同一个数据集的曲线图和枝干图：

```
t = 0:pi/20:2*pi;
y = exp(sin(t));
plotyy(t,y,t,y,'plot','stem')
```

绘图结果见图 4-6。

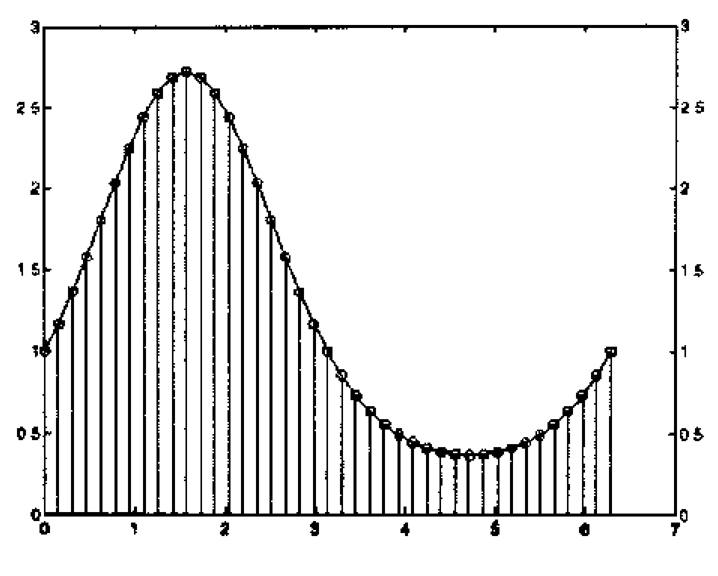


图 4-6　双 y 轴图形绘制

4.1.2　图形的叠加

hold 指令能帮助用户在原有的图形上再添加图形。如果用户将 hold 设置为 on，则再一次调用 plot 函数绘图时，MATLAB 不是擦除原有的图形，而是在原有图形的基础上重新添加图形，如果有必要将会重新标度坐标轴。

例如，以下语句先创建一个半对数坐标图形，然后在该图形上再叠加一个线性图形。

```
semilogx(1:100,'+')
hold on
plot(1:3:300,1:100,'--')
hold off
```

虽然 MATLAB 对半对数坐标进行了重新标度，但是并没有将坐标轴变为线性坐标。绘制结果如图 4-7 所示。

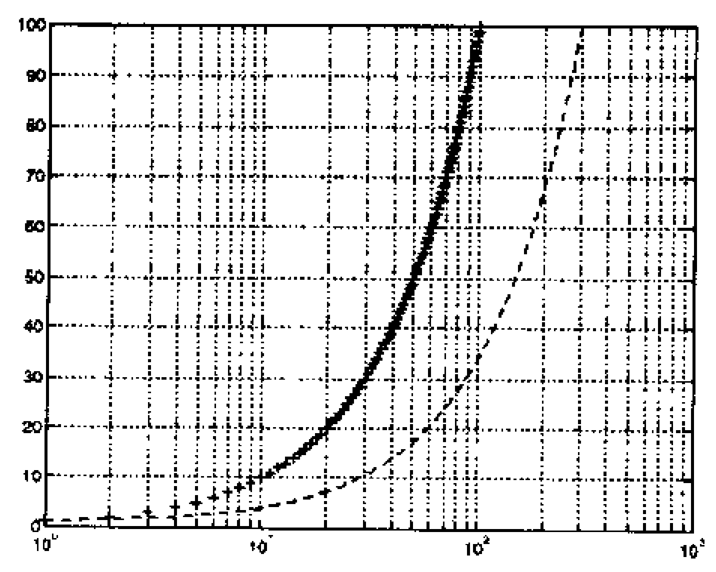


图 4-7　半对数与线性图形的叠加

使用 plotyy 函数也可以在同一幅图形中使用不同的坐标轴系统绘制不同范围的两个数据集。例如，以下语句将在同一个图形中绘制对数坐标下的指数函数图形和线性坐标下的正弦函数图形：

```
t = 0:900; A = 1000; a = 0.005; b = 0.005;
z1 = A*exp(-a*t);
z2 = sin(b*t);
[haxes,hline1,hline2] = plotyy(t,z1,t,z2,'semilogy','plot');
```

为了清楚地区分两个不同的坐标轴，需要给两个对坐标轴加上标签。首先给对数坐标轴加上标签：将对数坐标轴的句柄保存在haxes数组中，并设置该坐标轴为当前坐标轴，调用ylabel函数设置标签文本：

```
axes(haxes(1))
ylabel('Semilog Plot')
```

然后用相同方法给线性坐标轴加上标签：

```
axes(haxes(2))
ylabel('Linear Plot')
```

为了区分不同的函数曲线，使用点线来绘制第二条曲线：

```
set(hline2, 'LineStyle', '--')
```

绘制结果如图 4-8 所示。

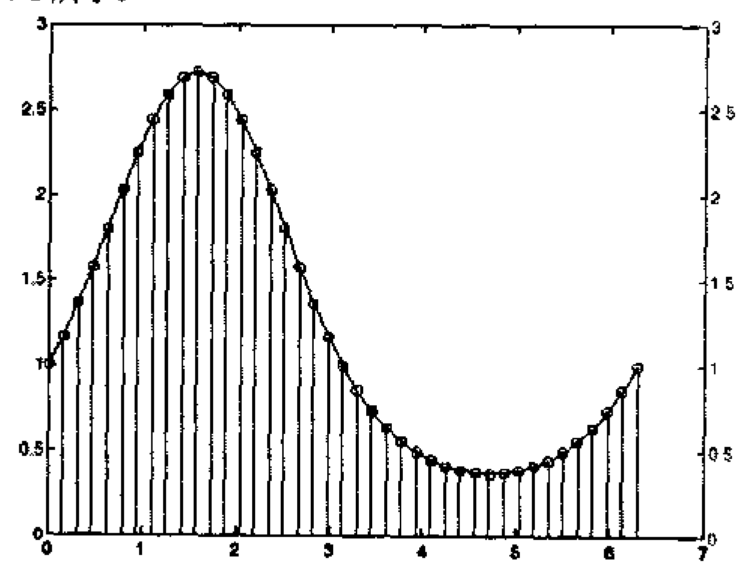


图 4-8 plotyy 函数实现的图形叠加

4.1.3 线型特征设置

通过将线型特征字符串传递给 plot 函数，可以给每一个数据集指定不同的线型，例如，如果上例使用以下语句：

```
t = 0:pi/100:2*pi;
y = sin(t);
y2 = sin(t-0.25);
```

```
y3 = sin(t-0.5);
plot(t,y,'-',t,y2,'--',t,y3,':')
```

将会产生与图 4-2 具有相同形状、不同线型的绘图结果（如图 4-9 所示）。

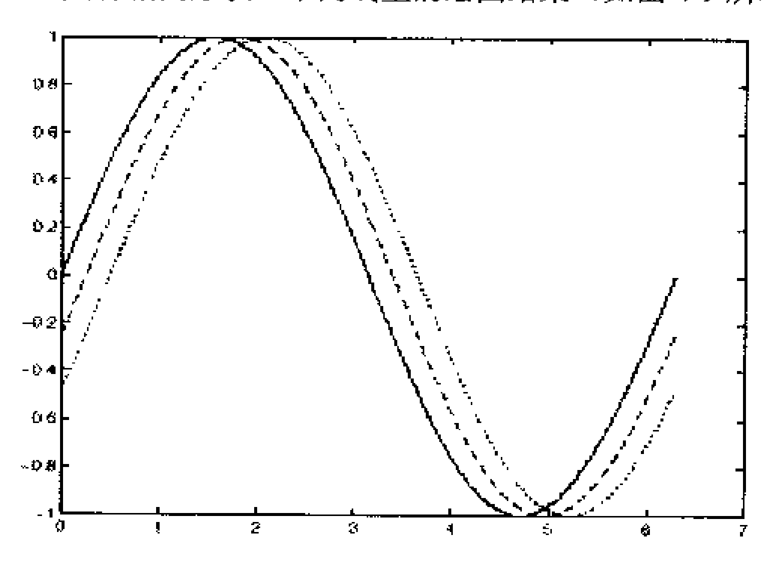


图 4-9　不同线型的多个曲线图形

其实，通过将字符串传递给绘图函数，用户不但可以给每个被绘制的向量指定线型，还可以指定其标示符形状和颜色。函数调用形式如下：

```
plot(x,y,'linestyle_marker_color')
```

其中 linestyle_marker_color 由以下几个部分组成（注意要将该字符串用单引号括起来）：

- 线型：例如点线、虚线等；
- 标示符类型：例如叉号、星号、圆圈等；
- 预定义颜色指示符：例如洋红色、蓝绿色、红色。

表 4-1 列出了常用的特征参数。

表 4-1　线型常用特征字符及其含义

线 型 特 征		标示符特征		预定义颜色	
字符	含义	字符	含义	字符	含义
-	实心线	+	加号	c	青色
--	虚线	o	圆形	m	洋红色
:	点线	*	星号	y	黄色
-.	点划线	x	差号	R	红色
none	隐藏线条	s	正方形	G	绿色
		d	钻石形	b	蓝色
		↑	上箭头	w	白色
		↓	下箭头	k	黑色
		→	右箭头		
		←	左箭头		
		p	五角星		
		h	六角星		
		none	不做记号		

例如：

```
plot(x,y,':squarey')
```

将绘制一条黄色的点线并使用正方形标记来标示曲线上每一个数据点。如果用户仅定义了标示符的类型而没有定义线型，那么绘图结果仅仅包含数据点的标示符。例如，下面在调用 plot 时仅仅给出颜色和标示符：

```
x = 0:pi/15:4*pi;
y = exp(2*cos(x));
plot(x,y,'r+')
```

以上语句将生成图 4-10 所示仅有表示数据点的红色加号的图形。

用户可以以任意顺序定义线条特征字符串，例如字符串：

'go--'

定义一个圆形（o）标示的点线（--），标示和线条都是绿色的（g）。

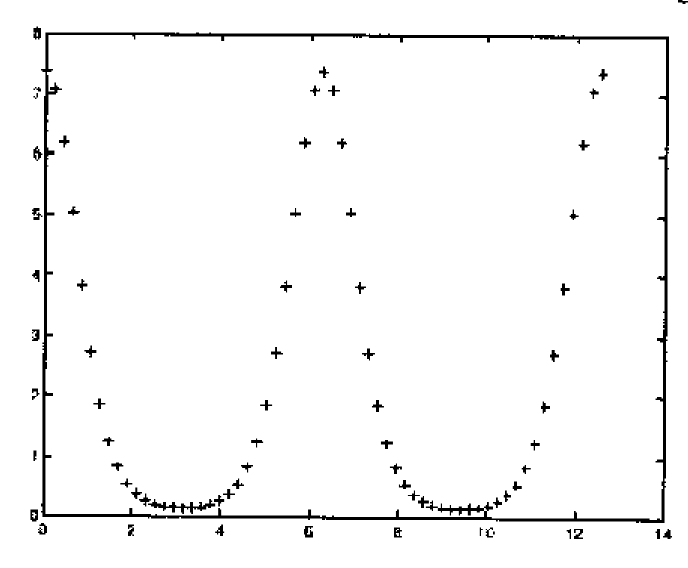


图 4-10 仅有标示符无线条的图形

另外，用户还可以通过定义线条的以下属性来控制线型特征：

- LineWidth：以像素点为单位定义线条的宽度；
- MarkerEdgeColor：指定标示符的颜色或指定填充型标示符的边缘颜色（圆形、正方形、钻石形、五角星形、六角星形和矩形）；
- MarkerFaceColor：指定填充型标示符的填充颜色；
- MarkerSize：以像素点为单位指定标示符的大小。

例如：

```
x=-pi:pi/10:pi;
y = tan(sin(x))-sin(tan(x));
plot(x,y,'rs','LineWidth',2, 'MarkerEdgeColor','k',...
    'MarkerFaceColor','g','MarkerSize',10)
```

将产生具有以下特征的图形（如图 4-11 所示）：一条带有正方形标示符的红色点线；线宽为两个像素点；标示符的边缘颜色为黑色；标示符表面颜色为绿色；标示符的尺寸设为 10 个像素点。

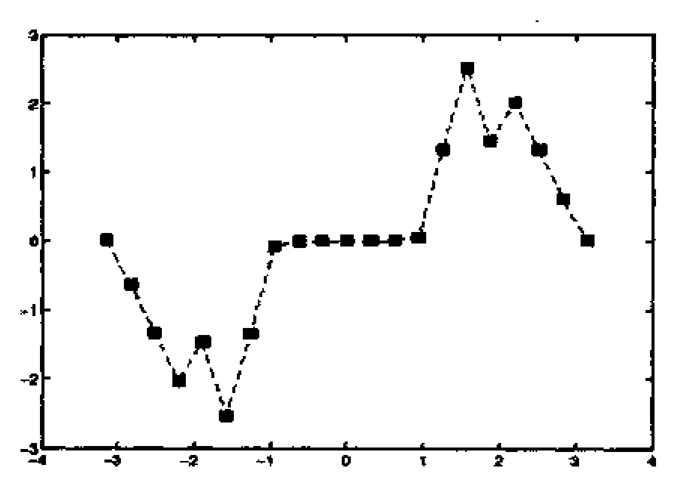


图 4-11 使用属性改变线型特征

如果没有给不同的数据集指定不同的显示颜色，或在使用黑白图形的情况下，通过定义不同的线型和标示符使得用户能够区分不同的数据集。

以上曾经介绍过，缺省情况下 MATLAB 使用不同的颜色区分不同的曲线。事实上，通过设置坐标轴的属性 LineStyle 组态 MATLAB，还可以使 MATLAB 在绘制多条曲线时使用不同的线型而不是颜色来区分不同的曲线。例如，以下语句将设置线型和颜色的缺省值：

```
set(0,'DefaultAxesLineStyleOrder',{'-o',':s','--+'})
set(0,'DefaultAxesColorOrder',[0.4,0.4,0.4])
```

然后使用设置好的缺省值绘制图形：

```
x = 0:pi/10:2*pi;
y1 = sin(x);
y2 = sin(x-pi/2);
y3 = sin(x-pi);
plot(x,y1,x,y2,x,y3)
```

由于将颜色的缺省值设置为统一的灰色，所以绘图时 MATLAB 不再使用颜色来区分不同的线条，而是通过设置的线型循环（实线圆形标示符、点线方型标示符、虚线加号标示符）来区分曲线。绘图结果见图 4-12。

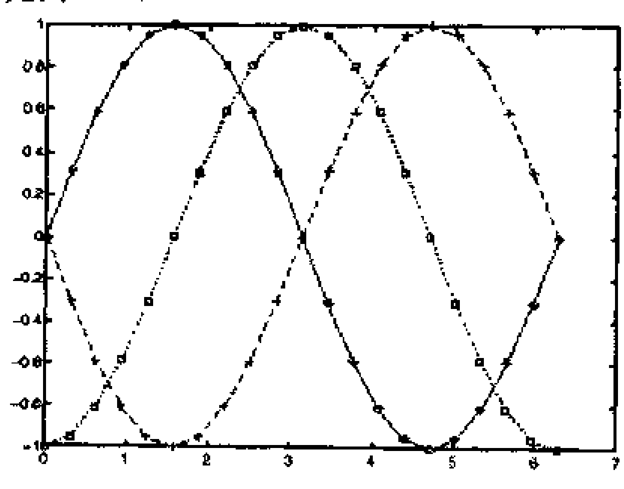


图 4-12 使用线型区分不同曲线

用户的这种缺省设置在 MATLAB 的整个运行期间内都有效。如果用户希望在 MATLAB 运行期间修改缺省值，使用保留字 remove：

```
set(0,'DefaultAxesLineStyleOrder','remove')
set(0,'DefaultAxesColorOrder','remove')
```

4.1.4　设置坐标轴属性

当用户创建图形时，MATLAB 根据数据范围自动选择坐标轴的范围和比例尺。用户也可以自己指定坐标轴的范围和刻度，以下命令可以用来设置坐标轴的这些属性：

- axis：设置影响当前坐标轴中对象的数值；
- axes：根据指定特性创建新的坐标轴对象（注意与 axis 区别）；
- get 和 set：使用户能够询问和设置已存在坐标轴对象的多种属性；
- gca：返回当前坐标轴的句柄。

下面介绍 axis：

axis 使用一个包含四个元素的输入向量来设置新的坐标轴范围：

```
axis([xmin,xmax,ymin, ymax])
```

注意 xmin 和 ymin 必须小于 xmax 和 ymax。

如果用户希望仅设置坐标轴的最大或最小值，另外几个限制由 MATLAB 自动设置，那么将需要 MATLAB 自动设置的参数用 Inf 或-Inf 来代替。例如，图 4-13 是使用 MATLAB 缺省坐标轴范围绘制的图形。

图 4-14 与图 4-13 曲线相同，但用户指定 x 坐标轴最大值，MATLAB 自动选择 x 轴最小值：

```
axis([-Inf 5 2 2.5])
```

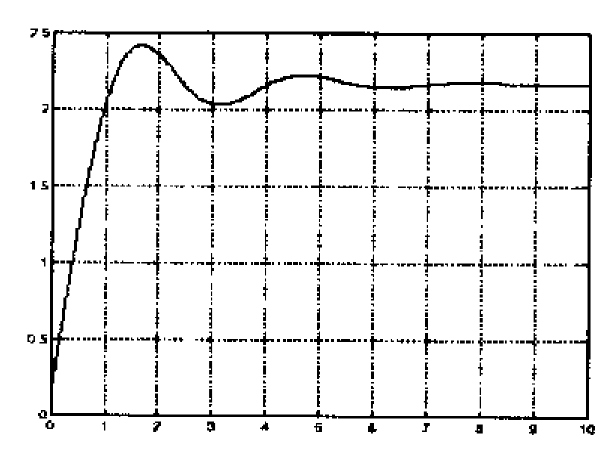


图 4-13　MATLAB 缺省坐标绘制图形

图 4-14　自定义坐标轴最大值绘制图形

用户可以通过设置坐标轴的 XTick 和 YTick 属性来改变坐标轴的刻度。定义一个逐渐增长的向量来定义坐标轴的刻度，向量元素的间隔可以不同。例如，对图 4-14 中的图形使用自己定义的 y 坐标轴刻度：

```
set(gca,'ytick',[2 2.1 2.2 2.3 2.4 2.5])
```

可得如图 4-15 所示的图形。

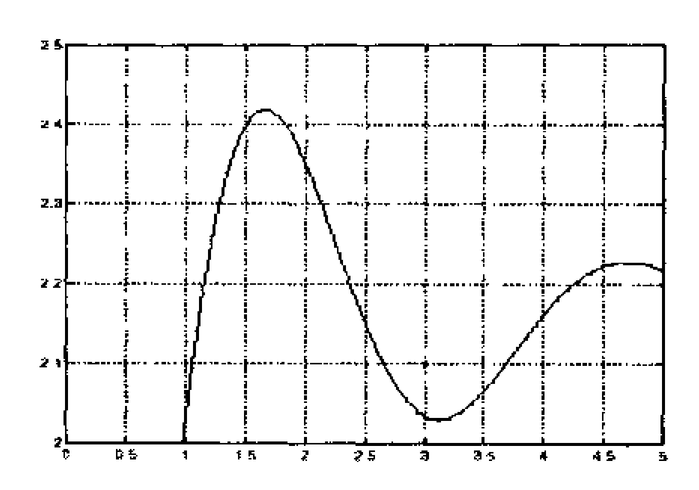


图 4-15　自定义 y 轴刻度绘制图形

如果用户定义的坐标轴刻度超过了坐标轴的范围，MATLAB 将会忽略这些刻度，也就是说，指定坐标轴的刻度不会改变坐标轴的范围。

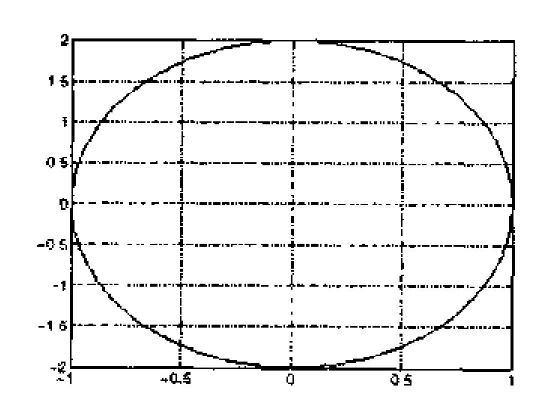
缺省情况下，MATLAB 在一个与图形窗口外观比例相同的矩形坐标轴系统中显示图形，这样能够使绘图空间得到最合理的应用。MATLAB 通过使用 axis 命令来实行外观比例控制。例如：

```
t = 0:pi/20:2*pi;
plot(sin(t),2*cos(t))
grid on
```

将产生一个使用缺省外观比例的图形（图 4-16 左），而命令：

```
axis square
```

将使 x 轴和 y 轴的长度相等（图 4-16 右）。



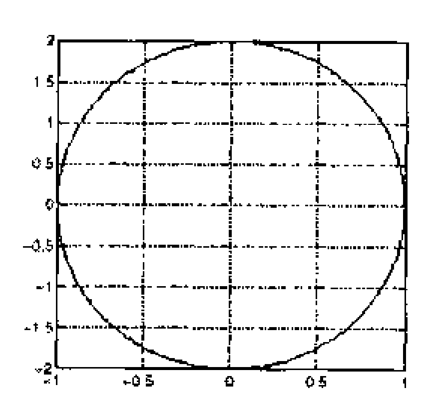
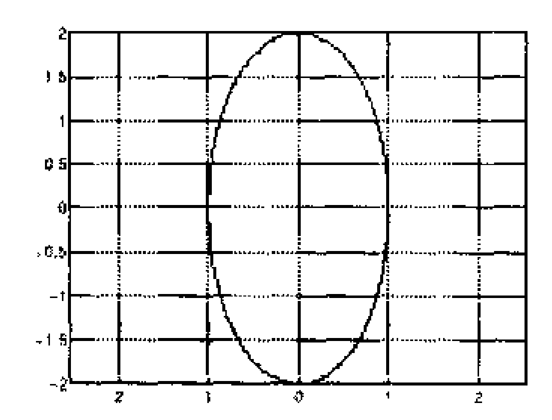


图 4-16　缺省和自定义坐标轴外观比例图形

在方形坐标轴中，x 轴的一个数据单位等于 y 轴两个数据单位。如果希望 x 轴和 y 轴数据单位等长，使用以下命令：

```
axis equal
```

该命令将会产生一个与图 4-16 外形相同但坐标轴比例尺不同的矩形坐标轴。如果希望坐标轴的形状能够与绘制数据范围相符，可以联合使用 tight 和 equal，其绘图结果参见图 4-17。



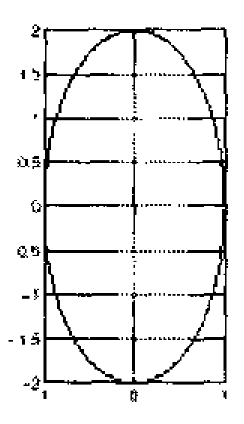


图 4-17 axis equal 和 axis equal tight 图形

4.1.5 图形窗口设置

MATLAB 直接将图形输出到一个不同于命令窗口的界面中，MATLAB 称这个窗口为图形窗口（figure）。这个窗口的特征由计算机图形操作系统和 MATLAB 图形窗口属性共同控制。如果当前不存在图形窗口，MATLAB 绘图函数将自动创建一个新的图形窗口；如果存在，那么将使用当前的图形窗口。图形窗口由 figure 函数创建，调用一次 figure 函数将创建一个新的图形窗口，并将该窗口设置为当前窗口。如果用户将某个图形窗口的句柄作为参数传递给 figure 函数：

```
figure(h)
```

那么该窗口将被设置为当前窗口。

通过使用 subplot 函数，用户可以在同一个图形窗口中绘制多个图形。Subplot(m,n,i)首先将整个图形窗口划分为 m×n 个子图形矩阵，然后选择第 i 个子图形作为当前图形。子图形按行从左到右编号。

例如，以下语句将在一个图形窗口的四个不同的子区域中绘制数据：

```
t = 0:pi/20:2*pi;
[x,y] = meshgrid(t);
subplot(2,2,1)
plot(sin(t),cos(t))
axis equal
subplot(2,2,2)
z = sin(x)+cos(y);
plot(t,z)
axis([0 2*pi-2 2])
subplot(2,2,3)
z = sin(x).*cos(y);
plot(t,z)
axis([0 2*pi-1 1])
subplot(2,2,4)
```

```
z = (sin(x).^2)-(cos(y).^2);
plot(t,z)
axis([0 2*pi-1 1])
```

显示结果如图 4-18 所示。

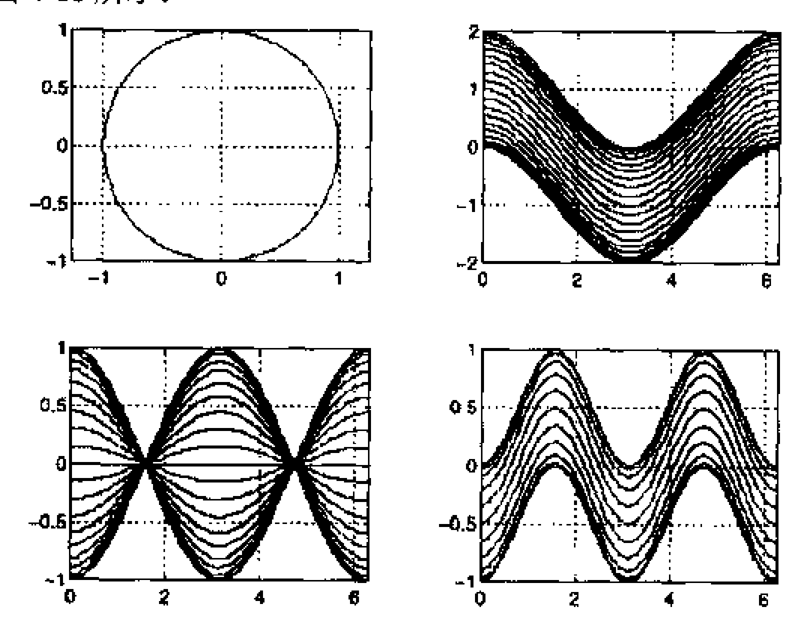


图 4-18　同一个窗口中绘制多个图形

每一个子区域图形都包含自身可以独立控制的坐标轴，用户可以设置和修改每个子图形坐标轴的属性。

由 subplot 定义的最后一个坐标轴是当前坐标轴。如果用户希望访问先前定义的坐标轴（例如给该坐标轴添加一个标题）那么首先要将该坐标轴设置为当前坐标轴。有三种方法可以将一个坐标轴设置为当前坐标轴：

- 一是用鼠标点击坐标轴所在的子图形；
- 二是调用 subplot，使用第三个参数来指定。例如：

```
subplot(2,2,2)
title('Top Right Plot')
```

将给图形窗口右上角的子窗口坐标轴添加一个标题；

- 三是使用坐标轴句柄调用 subplot。用户可以使用以下语句来获得所有子图形坐标轴的所有句柄：

```
h = get(gcf,'Children')
```

MATLAB 将返回所有子图形的坐标轴句柄，最后创建的最先返回。也就是说，如果将图形窗口分为 2×2 个子窗口，则 h(1)是子图形 224 的句柄，h(2)是子图形 223 的句柄，依此类推。如果希望将子图形 223 的坐标轴设置为当前坐标轴，使用语句：

```
subplot(h(2))
```

缺省的图形窗口颜色配置方法将为不同图形函数生成的图形实现良好的对比度和可见度。颜色配置将决定窗口的背景颜色、坐标轴的背景颜色、坐标轴线型和标签、绘图线条和曲面边缘的颜色以及其他影响外观的属性。

colordef 函数使用户既能够选择预定义的颜色配置，也能修改颜色。colordef 预定义了

三种颜色配置：

- colordef white：设置坐标轴背景颜色为白色，图形窗口背景颜色为灰色，曲面边缘颜色为黑色，还定义了其他一些绘图颜色和属性的数值；
- colordef black：设置坐标轴背景颜色为黑色，窗口背景颜色为暗灰色，曲面边缘颜色为黑色，以及其他相关属性；
- colordef none：设置与 MATLAB 4 相匹配的颜色。

用户可以通过观察 colordef.m 文件的源代码来查看 colordef 所定义的属性。

4.1.6 其他图形格式

当创建描述性图形时，用户可能会希望为图形添加一些注释来帮助用户理解数据。MATLAB 提供了以下机制来满足这个要求：

- 在坐标轴顶端添加一个标题；
- 添加一个图例；
- 添加坐标轴标签；
- 在图形的任意位置添加箭头和线条；
- 添加基本数据统计绘图，例如最大值、最小值和均值。

下面一一介绍这些添加方法。

1. 添加标题

标题是一个位于坐标轴顶端的文本字符串，一般用来说明图形的主题。可以通过 Insert 菜单的 Title 选项来添加一个图形标题，也可以在绘图编辑模式下通过改变坐标轴属性来添加标题，还可以使用 title 函数来添加。这里仅介绍修改坐标轴属性和 title 函数这两种添加方法。

第一种方法首先要启动绘图编辑模式，然后双击图形中的坐标轴对象打开属性编辑器。点击属性编辑器的 Labels 面板，在标题文本输入框中键入用户标题文本，应用后就完成了标题的添加。

如果需要在 MATLAB 命令行或 M 文件中中添加标题，可以使用第二种添加方法：titil 函数。例如，以下代码将为当前坐标轴添加一个标题并设置 FontWeight 属性为 bold（字体为粗体）。

```
title('Lotka-Volterra Predator-Prey Population Model'),'FontWeight','bold')
```

同样可以使用 set 函数来编辑命令行或 M 文件中的标题。

2. 添加图例

图例可以用来识别图形中绘制的不同数据集。使用 legend 函数来添加或编辑图例，例如，以下语句将给当前的坐标轴添加一个图例：

```
legend('Y1 Predator','Y2 Prey')
```

legend 函数还可以设置图例的位置等其他属性。

在图形编辑模式下，可以使用属性编辑器来编辑图例的外观并用鼠标将图例拖到用户所需的位置。如果图形编辑模式没有启动，可以右击图例对象，然后在弹出菜单中选择 Unlock Axes Position 选项，用鼠标的任意键拖动图例对象到所需位置。在这种情况下也可以使用属

性编辑器来编辑图例的文本属性。由于 Insert 菜单使用方法与编辑模式下添加和编辑坐标轴对象的方式基本相同，所以这里不再赘述，以下仅介绍各种对象的函数添加方法。

3. 添加坐标轴标签

坐标轴标签的创建和编辑同样有三种方式：Insert 菜单方式、图形编辑方式和函数方式。用户可以使用 xlabel、ylabel、zlabel 三个函数来分别添加或编辑 x、y、z 轴的标签。例如：

```
xlabel('t = 0 to 2\pi','FontSize',16)
ylabel('sin(t)','FontSize',16)
```

4. 添加自由格式的文本注释

使用text或gtext函数可以在图形的任意位置添加文本注释。如果使用text函数，用户必须使用x、y坐标值定义文本在图形中的位置。例如：

```
str1(1) = {'Many Predators;'};
str1(2) = {'Prey Population'};
str1(3) = {'Will Decline'};
text(7,220,str1)
str2(1) = {'Few Predators;'};
str2(2) = {'Prey Population'};
str2(3) = {'Will Increase'};
text(5.5,125,str2)
```

上例也说明了如何使用单元数组来创建多行文本注释。

用户还可以计算文本注释在图形中的位置。以下语句将在图形中的三个数据点处添加文本注释：

```
text(3*pi/4,sin(3*pi/4), '\leftarrowsin(t) = .707','FontSize',16)
text(pi,sin(pi),'\leftarrowsin(t) = 0', 'FontSize',16)
text(5*pi/4,sin(5*pi/4),'sin(t) =-0.707\rightarrow', ...
    'HorizontalAlignment','right', 'FontSize',16)
```

这里要注意，MATLAB 根据 HorizontalAlignment 属性值将文本注释字符串添加到点[5*π/4,sin(5*π/4)]的左边。HorizontalAlignment 和 VerticalAlignment 属性控制文本字符串的放置，缺省时 HorizontalAlignment 为 left，VerticalAlignment 为 middle。当属性 VerticalAlignment 取值为 middle，属性 HorizontalAlignment 取不同值，或属性 HorizontalAlignment 为 left，属性 VerticalAlignment 取不同值时的文本外观如图 4-19 所示。

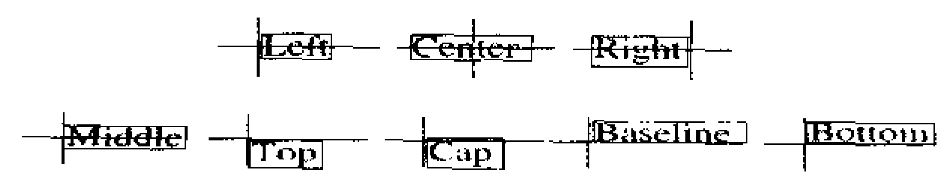


图 4-19　HorizontalAlignment 和 VerticalAlignment 不同取值时的文本外观

5. 添加箭头和线条

箭头和线条的添加都是伴随着文本注释进行的，一般不单独添加这两个内容。

6. 添加基本数据统计绘图

举例来说明数据统计绘图的使用方法。

首先要装入数据并绘制，例如美国人口普查结果显示的美国历史人口数据：

load census

plot(cdate,pop,'+')

然后在 Tools 菜单下选择 Data Statistics 选项，显示如图 4-20 所示的数据集统计结果。在该对话框中选择所需的数据统计结果，图形中就会出现该统计结果图形。然后用户可以为该图形添加一个图例。

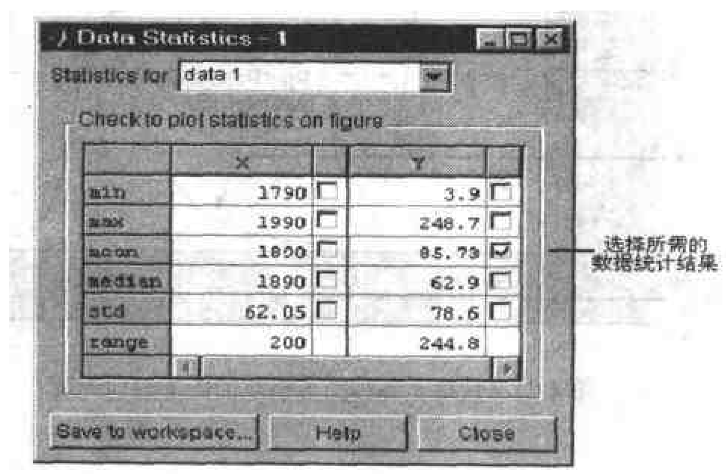


图 4-20 数据统计结果对话框外观

用户可以将这些统计结果保存在 MATLAB 工作平台中以备以后的计算使用。保存过程如下：

点击 Save to Workspace 按钮，在 Save Statistics to Workspace 对话框中选择需要保存的统计结果并指定将结果保存到哪个变量中。假设将美国人口历史记录的 x 数据集统计结果保存到一个结构体变量 census_dates 中，则该变量的内容如下：

```
census_dates =
        min: 1790
        max: 1990
        mean: 1890
        median: 1890
        std: 62.0484
        range: 200
```

事实上统计对话框的所有工作都是通过基本数据统计绘图函数实现的。基本数据统计绘图函数仅适用于二维图形，常用的有：

max：求数据集的最大值；

min：求数据集的最小值；

mean：求数据集中所有数据的平均值；

median：求数据集中间数值；

n/a：数据集中最大值与最小值之间的间隔（该函数不产生绘图结果）；

std：求数据集的标准方差。

图 4-21 是一个使用上述 6 种机制得到的图形。

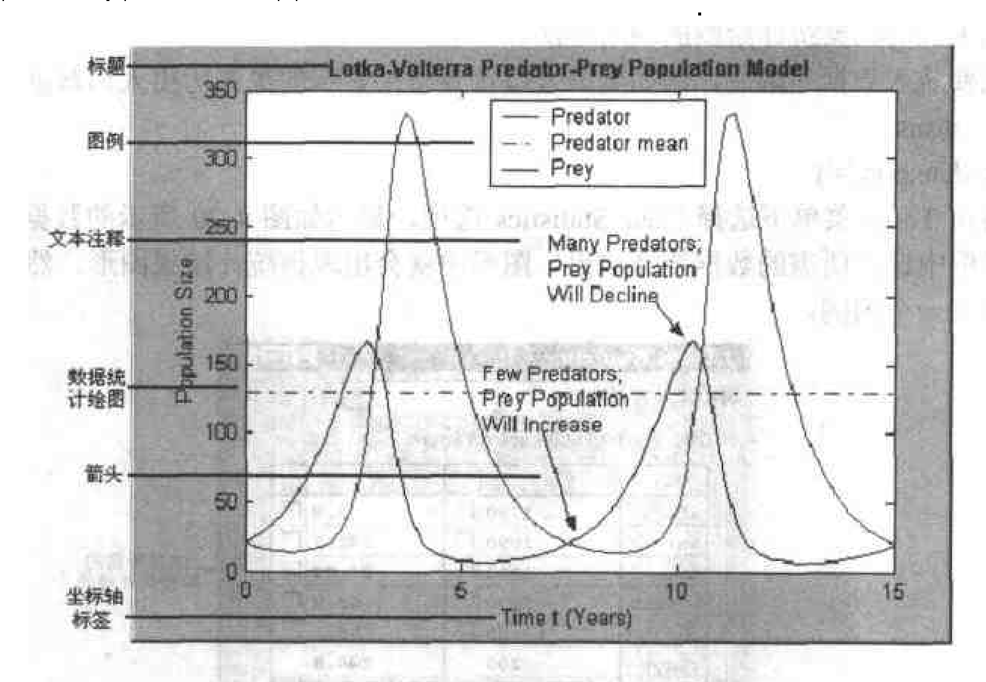


图 4-21　使用了各种图形格式的图形窗口外观

4.2　图像的显示和处理

4.2.1　MATLAB 图像简介

MATLAB 为各种图形文件格式的图像提供了读写和显示函数。一旦 MATLAB 实现了某种图形文件格式的图像的显示工作，该图像就会与 MATLAB 自身生成的图像一样成为一个图形对象。MATLAB 支持以下格式的图像：BMP、HDF、JPEG、PCX、PNG、TIFF、XWD。如果用户希望了解更多的支持信息，可以查看 imread 和 imwrite 文件的源代码。

MATLAB 将大多数的图像存储为矩阵（二维数组），矩阵的每一个元素对应于图像的一个像素。例如，如果一幅图像包含 200×300 个不同色点，那么 MATLAB 将该图像存储为一个 200×300 的矩阵。某些图像（例如 RGB 图像）需要一个三维数组来存储，该数组的第一维表示像素的红色强度，第二维表示绿色强度，第三维表示蓝色强度。MATLAB 支持三种不同的数字图像数据类型（即图像存储类型）：双精度浮点数、16 位无符号整数和 8 位无符号整数。图像文件存储格式不同，图像显示函数解释数据的方式也不同。事实上图像数据完全可以存储为双精度格式，但为了节约内存，MATLAB 为图像数据提供了 8 位和 16 位存储格式。

在 MATLAB 中，一幅图像由一个数据矩阵和一个可能存在的颜色映射矩阵组成。根据 MATLAB 对矩阵数据的理解方式的不同可以将图像分为以下三种类型：

- 索引图像；
- 亮度（灰度）图像；

● RGB图像。

下面将对这三种图像进行一一介绍。

1. 索引图像

一幅索引图像由一个数据矩阵 X 和一个颜色映射矩阵（即调色板）map 组成。map 是一个 m×3 的浮点数数组，其元素的取值范围为 0 到 1。Map 数组的每一行指定一个点的红色、绿色和蓝色颜色分量。索引图像直接将像素数值映射为调色板的数值，也就是说，直接将矩阵 X 的数值作为 map 的下标，得到的数值就是像素的颜色值。例如，矩阵 X 某元素的数值为 1 则表示对应像素的颜色取调色板的第一个颜色，为 2 则取第二个颜色，依此类推。用户可以使用以下语句来显示一幅索引图像：

```
image(X);
colormap(map)
```

图像矩阵数值与调色板之间的关系依赖于图形矩阵的种类。如果图形矩阵数据是双精度存储类型的，那么数值 1 就代表颜色索引表的第一项，数值 2 就代表第二项，依此类推；如果图像矩阵数据是 8 位或 16 位无符号整数存储类型的，那么将产生一个偏移量：数值 0 表示调色板的第一项，数值 1 表示第二项，依此类推。有时偏移量也被用于最大化系统对图形文件颜色的支持。

图 4-22 说明了一幅索引图像的结构，图像的像素由整数来表述，这些整数表示该像素颜色在调色板中的位置。可以看出该图像文件的数据是采用双精度存储类型的（无偏移量）。

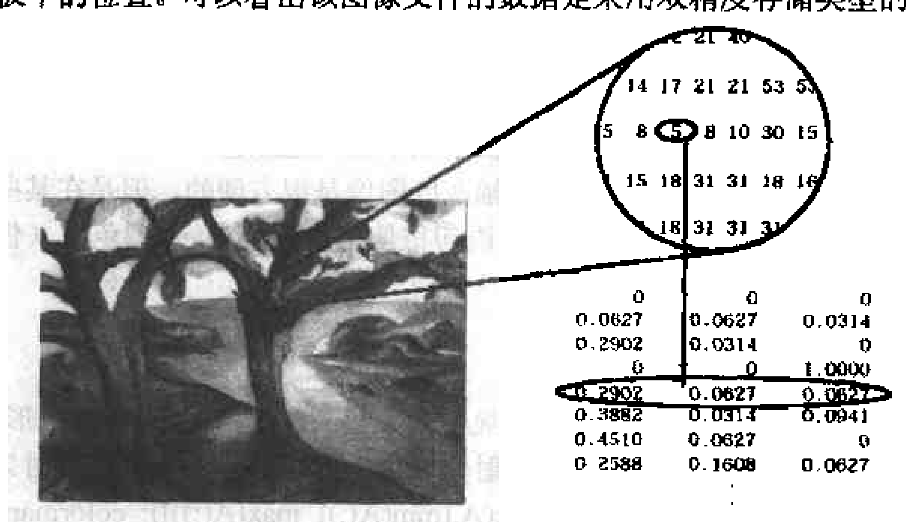


图 4-22 索引图像数据与调色板映射关系

调色板通常是伴随着索引图像数据一同存储的，在调用 imread 函数时，调色板会和图像一起自动装载。

2. 灰度图像

灰度图像是一个数据矩阵 I，该矩阵的数值代表点在一定范围内的灰度(亮度)。MATLAB 使用一个单独的矩阵来存储灰度图像，该矩阵的每个元素都对应一个像素点。灰度图像的数据矩阵可以是双精度存储类型的，也可以是 8 位或 16 位存储类型的。虽然灰度图像并不保存调色板矩阵，但是 MATLAB 仍使用系统指定的灰度扫描调色板来显示该图像。从本质上说，MATLAB 控制灰度图像的方法与索引图像相同。图 4-23 描述了一个双精度存储类型的灰

度图像。

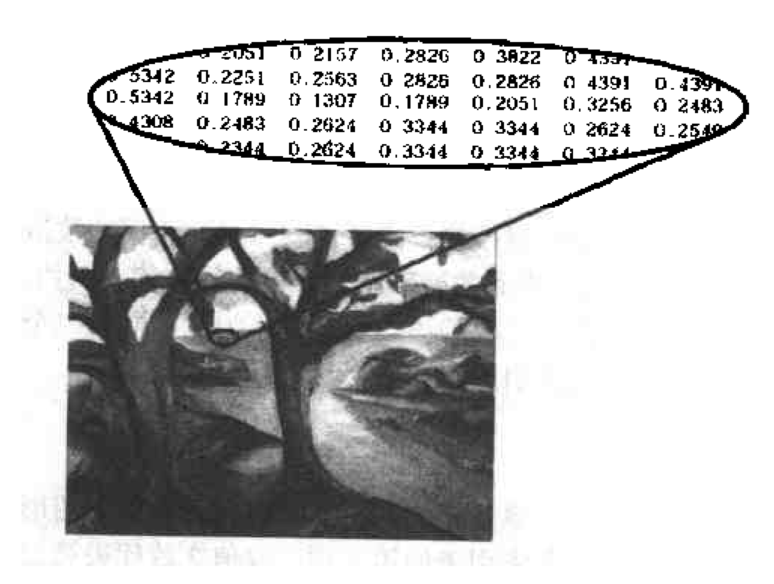


图 4-23　双精度灰度图像与数据

通常使用 imagesc 函数来显示一幅灰度图像，该函数允许用户指定灰度范围。imgaesc 使用两个输入参数来显示灰度图形。例如：

imagesc(I,[0 1]);

colormap(gray);

imagesc 的第二个参数指定了图像需要的灰度范围，imagesc 通过将该范围内的第一个数值映射到灰度扫描调色板的入口、第二个值映射到调色板的最后一个入口来显示矩阵 I。其他在该范围内的数值将线性地映射为调色板中剩下的其他颜色。

尽管使用一个灰度扫描调色板来显示一幅灰度图像是很方便的，但是在某些情况下也会使用其他的调色板来显示图像。例如，以下语句将使用名为 winter 调色板（仅包含蓝色和绿色）来描述灰度图像 I：

imagesc(I,[0 1]);

colormap(winter);

如果希望将一个取值范围不定的矩阵 A 显示为一幅灰度图像，使用单参数形式的 imagesc 函数。MATLAB 将矩阵 A 数据的最小值映射为调色板的第一项，最大值映射为最后一项。例如：imagesc(A); colormap(gray)与 imagesc(A,[min(A(:)) max(A(:))]); colormap(gray)是等效的。

3. RGB（真彩）图像

RGB 图像（也称为真彩图像）是一个 m×n×3 的数组 RGB，该数组的每一个元素表示一个像素的红色、绿色和蓝色颜色分量。RGB 图像不使用调色板，每一个像素的颜色由相应位置的像素红绿蓝颜色分量的组合决定。图形文件格式将 RGB 图像存储为 24 位图像，红绿蓝成分各占一个字节。这样的图像可能会存在 1,6000,000 种颜色，可以基本复制物体的真实颜色，故称之为真彩图形。

一个 RGB 数组也可以是双精度或 8 位、16 位的。在双精度类型中，每一个颜色成分的取值范围从 0 到 1。颜色成分为(0,0,0)的像素显示为黑色，颜色成分为(1,1,1)的像素显示为

白色。每个像素的三种颜色成分顺序地存储在数据数组的三维空间中。例如，像素(10,5)的三种颜色成分分别储存在 RGB(10,5,1)、RGB(10,5,2)和 RGB(10,5,3)中。

使用 image 函数显示一个真彩图像：

image(RGB)

图 4-24 是一幅双精度类型的 RGB 图像。

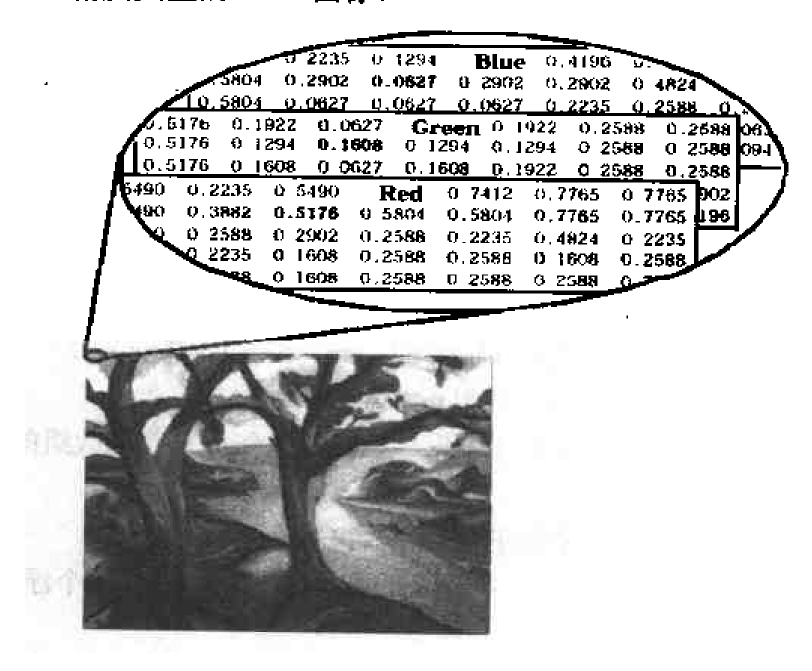


图 4-24　双精度 RGB 图像

为了确定位于(2,3)的像素的颜色，需要观察(2,3,1:3)RGB 数据组的数值。假设这些数值分别为：0.5716，0.1608，0.0627，那么像素的颜色就是：

[0.5716　0.1608　0.0627]

如果 MATLAB 在一台没有真彩图像硬件支持的计算机上工作，那么 MATLAB 将选择最相近的颜色并使用抖动技术来显示类似的图像。

4. 8 位和 16 位图像操作

MATLAB 通常使用双精度浮点数（64 位）来工作，但是，为了减少图像操作所需的内存，MATLAB 也可以使用 uint8 或 uint16（8 位或 16 位无符号整数）数值类型来存储图像。数据矩阵类型为 uint8 的图像称为 8 位图像，为 uint16 的图像称为 16 位图像。

（1）8 位和 16 位索引图像：如果矩阵 X 的数据类型是 uint8 或 unit16，在将这些数值映射到调色板之前，首先要做一次偏移。image 函数自动提供合适的偏移计算，所以无论矩阵 X 是双精度还是 uint8 或 uint16 类型，图像的显示方法都是一样的：

image(X);

colormap(map);

之所以要对 uint8 和 uint16 数据进行偏移计算是为了支持常用的 256 色标准图形文件格式调色板。由于这种偏移，用户在将 uint8 或 uint16 索引图像转换为双精度类型图像时必须给每个数据值加 1：

X64 = double(X8) + 1;

或者：

X64 = double(X16) + 1;

相应的，如果要将双精度图像转换为 uint8 或 uint16，必须给每个数据减 1：

X8 = uint8(X64 – 1);

或者：

X16 = uint16(X64 – 1);

通过使用内存利用率更高的 uint8 和 uint16 数组，用户可以在 MATLAB 中进行更为有效的图像操作。这里要注意，MATLAB 的许多数学操作对 uint8 和 uint16 数据类型都是不适用的。支持 uint8 和 uint16 数据类型的数学函数有：conv2、convn、fft2、fftn 和 sum。这些函数对这两种数据类型操作的结果都是双精度浮点数。如果用户试图使用不支持这两种数据类型的函数来操作这两种类型的数组时，MATLAB 将报错。MATLAB 图像处理工具箱中的大多数函数支持 uint8 和 uint16 数据类型，因此，如果用户希望获得良好的图像处理效果，最好在计算机上安装图像处理工具箱（Image Proce- ssing Toolbox）。

另外，MATLAB 还支持以下 uint8 和 uint16 数组的操作：

- 使用 reshape、cat、permute、[ ]和 ’ 函数或操作符来实现数组的再成型、重排以及连接；
- 使用 save 和 load 将数组保存在 MAT 文件中；
- 使用 find 函数来搜索数组中非零元素的下标，返回值总是一个浮点型数组；
- 各种关系操作符。

（2）8 位和 16 位灰度图像：如果希望使用灰度调色板显示一幅 8 位灰度图像，可以使用以下语句：

imagesc(i,[0 255]);

colormap(gray);

这里要注意的是，由于双精度图像的数据范围通常是[0，1]，而 uint16 是[0，65535]，所以如果希望将一个双精度类型的图像转换成 uint16 类型，那么必须首先给每个元素乘以 65535：

I16 = uint16(round(I64*65535));

相应的，为了将一个 uint16 类型的图像转换为双精度类型，必须给每个数据除以 65535：

I64 = double(I16)/65535;

uint8 与双精度类型图像的转换方法与之类似。

（3）8 位和 16 位 RGB 图像：8 位 RGB 图像的数据成分是 0 到 255 范围内的整数，而不是 0 到 1 之间的浮点数，颜色成分为(255,255,255)的像素将显示为白色。为了将一个双精度类型的 RFB 图像转换为 uint8 类型，首先要给每个数值乘以 255：

RGB8 = uint8(round(RGB64*255));

转换为 uint16 则要乘以 65535：

RGB16 = uint16(round(RGB64*65535));

事实上，image 函数可以直接显示 8 位或 16 位图像而无需将它们转换为双精度类型，然而 image 函数解释这两种图像矩阵数据的方法略有不同。

4.2.2 图形图像的读写和查询

一般图形文件都是以一个包含格式信息的文件头开始，之后是可以作为连续数据流读写的图像数据，从本质上说，图形文件格式的图像与MATLAB其他矩阵的存储方式是不同的，甚至图像文件可以不作为一个矩阵来存储。因此，用户不能通过一般的MATLAB输入输出命令来对图形文件格式的图像进行装载和保存或进行图像的读写。

MATLAB提供一些特殊的函数实现图形格式图像数据的读写：使用imread函数来读一个图形文件格式的图像；使用imwrite来写一个图形文件格式的图像；如果要查询图形文件格式图像的信息，使用imfinfo。

1. 读图形图像

函数imread能够从任意一个MATLAB支持的图像文件中（事实上大多为8位）读取一幅图像。当MATLAB将图像读入内存后，MATLAB一般使用uint8来存储图像，只有PNC和TIFF格式的文件使用uint16来保存。对于索引图像而言，无论图像数据是uint8还是uint16，imread函数总将调色板读到一个双精度的矩阵中。

例如，使用imread读入一个ngc6543a.jpg文件的代码如下：

```
RGB = imread('ngc6543a.jpg');
```

使用imwrite来写一个小丑图像：

```
load clown
imwrite(X,map,'clown.bmp')
```

以上语句将创建一个BMP文件，然后将小丑图像保存在该文件中。

2. 写图形图像

当用户使用imwrite函数保存一幅图像时，MATLAB的缺省行为是将图像类型自动转换为uint8类型。大多数图像在使用时无需是双精度类型的，所以这一点在一般情况下符合用户对图像文件进行读写的要求。但是，对于RNG和TIFF文件来说，由于这两种格式支持的是16位数据，所以使用这两种格式时用户必须覆盖MATLAB在调用imwrite函数时的缺省行为。以下语句说明了如何使用imwrite写16位的RNG文件：

```
imwrite(I,'clown.png','BitDepth',16);
```

3. 获得图像文件有关信息

imfinfo函数使用户能够获得任意MATLAB支持格式的图形文件有关信息。信息的具体内容与文件格式有关，但是无论何种格式都包括以下几种信息：

- 文件名，如果该文件不在当前目录中，则还包括路径信息；
- 文件格式及其版本号；
- 文件修正数据；
- 文件大小（按字节计算）；
- 图像宽度和高度（以像素计算）；
- 每个像素所占的位数；
- 图像类型：索引、灰度或RGB。

4.2.3　图像显示

使用image或imagesc函数显示一幅图形文件格式的图像。例如：

```
figure('Position',[100 100 size(RGB,2) size(RGB,1)]);
image(RGB); set(gca,'Position',[0 0 1 1])
```

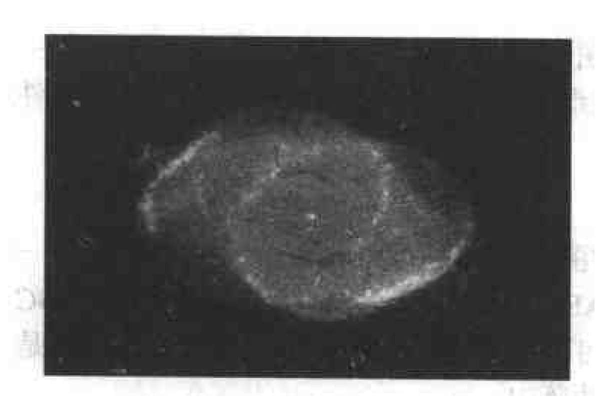


图 4-25　图像显示

以上语句将显示如图 4-25 所示的图像。

对于不同的图像格式，显示函数及其格式也不同。

对于索引图像，使用以下语句显示：

```
image(X);
colormap(map)
```

对于灰度图像，使用以下语句显示：

```
imagesc(I,[0 1]);
colormap(gray)
```

对于 RGB（真彩）图像，使用：

```
image(RGB)
```

image 函数使用缺省大小的图形窗口和坐标轴来显示图像，MATLAB 会对图像进行缩放以适合显示区域的要求。如果用户希望图像的外观比例符合图像数据矩阵的大小，最简单的方法就是使用 axis image。

例如，以下语句将使用缺省位置的图形窗口和坐标轴显示一个地球图像：

```
load earth
image(X);
colormap(map)
```

显示结果如图 4-26 所示，为了适应坐标轴的要求，该图像被拉长了。

以下命令将迫使该地球图像的外观比例变为 1:1（图 4-27）。

```
axis image
```



图 4-26　缺省设置下的图像显示效果

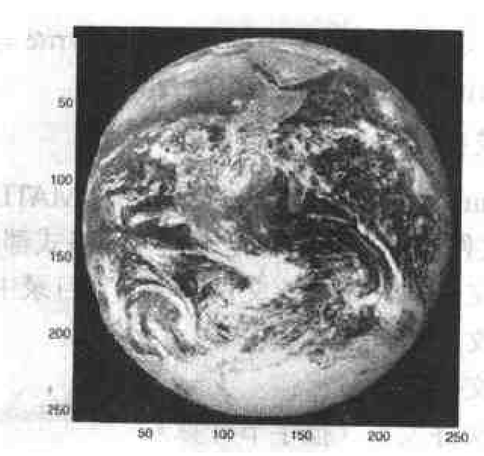


图 4-27　坐标轴比例为 1:1 的图形显示效果

axis image 命令是通过设置坐标轴对象的 DataAspectRatio 属性值为[1 1 1]来实现这一工作的。

有时用户可能会希望将图像数据矩阵中的每一个元素与屏幕上的每一个像素对应起来，这就必须重置图形窗口和坐标轴的尺寸。例如，以下语句将使地球图像数据矩阵的每一个像素与屏幕上的点相对应（图4-28所示）：

```
[m,n] = size(X);
figure('Units','pixels','Position',[100 100 n m])
image(X); colormap(map)
set(gca,'Position',[0 0 1 1])
```

图形窗口的位置属性（Position）由一个包含四个元素的向量（四个元素分别用来指定图形窗口位置和大小）决定。以上语句还将图形窗口的左下角设置在(100,100)处，使得窗口的长度和宽度符合图像的高度和宽度。



图4-28 指定窗口位置和坐标轴尺寸的图形显示效果

4.2.4 图像对象及其属性

调用image和imagesc函数将创建一个图像对象，该图像对象是坐标轴的子对象。与其他类型的图形对象一样，图像对象有许多可以设置的属性，可以通过属性设置使图像的显示符合屏幕的外观。图像对象最主要的属性有CData，CDataMapping，XData，YData和EraseMode。

CData包含图像的数据数组。例如，以下命令中，h是由image创建的图像对象的句柄，矩阵X和Y的内容是相同的：

```
h = image(X); colormap(map)
Y = get(h,'CData');
```

CData数组的维数将决定MATLAB显示图像时使用调色板与否。如果CData是二维数组，那么图像就是索引图像或灰度图像，在显示图像时需要使用调色板；如果CData是m×n×3维的数组，那么该图像为RGB图像，无需调色板。

CDataMapping控制决定图像是索引图像还是灰度图像。索引图像的CDataMapping属性值为direct，在这种情况下，CData数据将直接作为窗口调色板的下标。如果调用image函数时仅使用一个参数，那么CdataMapping就将被设置为direct：

```
h = image(X); colormap(map)
get(h,'CDataMapping')
ans =
    direct
```

显示灰度图像时，CDataMapping的属性值被设为scaled，此时CData的数据将线性地映射到调色板中，映射因数由坐标轴的CLim属性控制。imagesc函数将创建一个图像对象，该对象的CDataMapping属性被设置为scaled，同时调节图像对象父坐标轴的CLim属性。例如：

```
h = imagesc(I,[0 1]);
colormap(map)
get(h,'CDataMapping')
ans =
    scaled
```

```
get(gca,'CLim')
ans =
    [0 1]
```

XData和YData属性控制图像的坐标轴系统。对于一个m×n的图像，XData和YData的设置遵循以下原则：图像最左边列的x坐标为1；图像最右边列的x坐标为n；图像最上边列的y坐标为1；图像最下边列的y坐标为m。

例如，以下语句将产生如图4-29所示的图片：

```
X = [1 2 3 4; 5 6 7 8; 9 10 11 12];
h = image(X);
   colormap(colorcube(12))
xlabel x;
ylabel y;
```

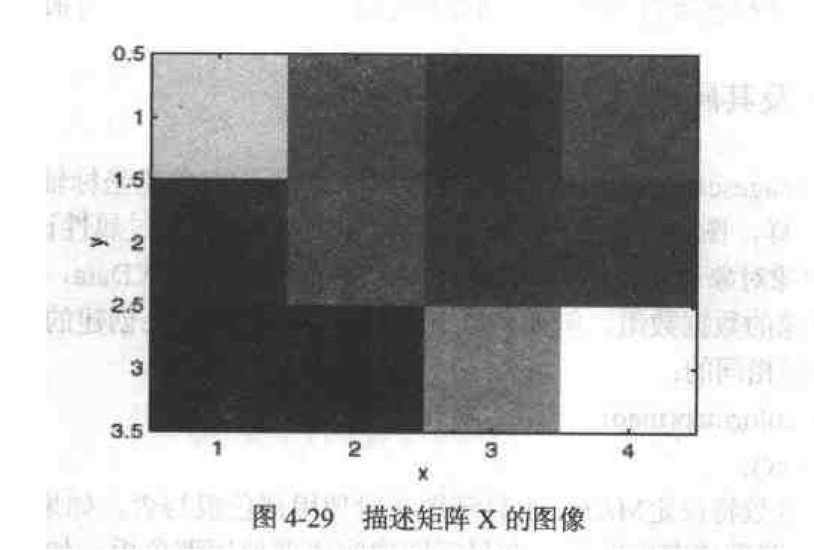


图 4-29　描述矩阵 X 的图像

以上语句生成的图像对象的XData和YData如下：

```
get(h,'XData')
ans =
     1 4
get(h,'YData')
ans =
     1 3; x; y; 1 2 3 4; 0.5; 1; 1.5; 2; 2.5; 3; 3.5;
```

用户可以覆盖这两个属性的缺省设置，使用自定义的坐标轴：

```
X = [1 2 3 4; 5 6 7 8; 9 10 11 12];
image(X,'XData',[-1 2],'YData',[2 4]);
colormap(colorcube(12))
xlabel x;
ylabel y
```

图 4-29 所示的图片将变为图 4-30 所示的外观。

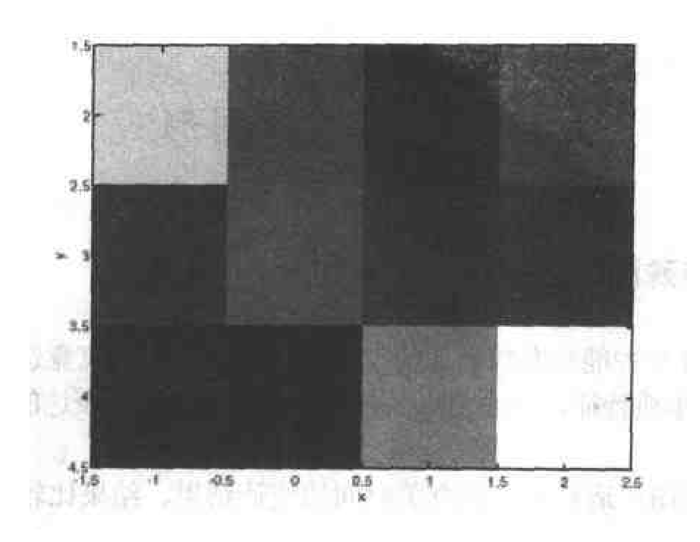


图4-30 改变XData和YData属性后图像显示效果

EraseMode：当CData发生变化时，该属性将控制MATLAB如何在屏幕上重画该图像。EraseMode的缺省值是normal，此时如果使用set命令来改变图像的CData，则在使用新的Cdata显示图像前，MATLAB将擦除原来的图像。这种方式不能快速平滑地显示一系列图像。如果将EraseMode设置为none，则当CData变化时，MATLAB不重画屏幕，这就使得图像能够快速平滑地重新显示。例如，如果有一个m×n×3×x的矩阵A，包含x个相同大小的不同RGB图像，那么用户可以使用以下语句动态平滑地显示这些图片：

```
h = image(A(:,:,:,1),'EraseMode','none');
for i = 2:x
set(h,'CData',A(:,:,:,i))
drawnow
end
```

有时用户会希望对图形进行一些uint8和uint16格式不支持的操作，为此要使用double函数将图形数据转换为双精度浮点数类型。例如：

```
BW3 = double(BW1) + double(BW2);
```

值得注意的是，数据类型间的转换将导致MATLAB及其工具箱对图像数据理解方式的改变。假如用户希望对一幅存储为索引图像的色彩图像进行滤波，首先必须将它转换为RGB格式，ind2rgb函数能够方便地实现这一操作。之所以要进行转换，是因为当用户对RGB图像进行滤波时，MATLAB将滤掉图像的颜色成分，但对一幅索引图像进行滤波只能滤掉图像矩阵的下标，因而产生无意义的结果。

用户也可以使用简单的数组基本操作进行某些转换。例如，如果希望将一个灰度图像转换为RGB图像，可以将原始矩阵的三个拷贝连接起来：

```
RGB = cat(3,I,I,I);
```

产生的RGB图像有明确的红色、绿色和蓝色矩阵。

有时用户会希望改变图像的图形文件格式来适应其他软件的要求，在MATLAB中这个目的很容易实现。假如需要将一个BMP图像my.bmp转换为PNG图像，只要使用imread装入该文件，指定文件装入格式为PNG即可：

```
imread(my.bmp,png);
```

4.3　特殊二维图形

4.3.1　MATLAB 特殊图形介绍

MATLAB 支持许多种能够有效表达信息的特殊图形。用户究竟选择何种图形很大程度上依赖于用户数据的本质特征，下面列出了各种特殊图形所擅长表达的数据类型，供读者参考。

- 直方图和面积图：适合于表达有关时间的统计结果、结果比较、显示总体中的个体分布情况；
- 饼形图表：适合于显示总体中的个体分布情况；
- 柱状图：适于显示数据值的分布情况；
- 枝干图和阶梯图：适于表示离散数据；
- 罗盘图、羽状图和轮廓图：适于显示向量的方向和大小；
- 等值线图：适于显示数据中的等值区域；
- 交互式绘图：使用户能够使用指针选择数据点进行绘图；
- 动画：通过一系列绘图给数据多增加了一维效果。

由于动画在上一章中已作过介绍，因而本节中仅介绍其他几种特殊图形。

4.3.2　直方图

直方图和面积图用来显示向量或矩阵数据。这两种图形在进行诸如观察某段时间的结果、比较两个不同数据集的结果以及表明个体在总体中是如何分布的这些情况下是非常有用的。直方图比较适合显示离散数据，面积图则适合显示连续数据。

MATLAB 有两个绘制二维直方图的函数：bar 和 barh，前者用来绘制竖直的直方图，后者用来绘制水平的直方图。缺省情况下，直方图将矩阵的每一个元素描述为一个直方条。二维直方图的直方条是分布在 x 轴上的，例如，定义一个简单矩阵 Y，使用 bar 函数的最简形式来绘制 Y 的直方图：

```
Y = [5 2 1
     8 7 3
     9 8 6
     5 5 5
     4 3 2];
bar(Y)
```

绘图结果见图 4-31。

由图 4-31 可见，直方图是一组丛生的均匀分布图形，每一丛代表矩阵的一行。

直方图还能够说明矩阵同一行中的元素在该行中是如何分布的，这种直方图称之为堆型直方图。

堆型直方图为矩阵的每一行显示一个直方条，该直方条分为n段，其中n表示矩阵的列数。

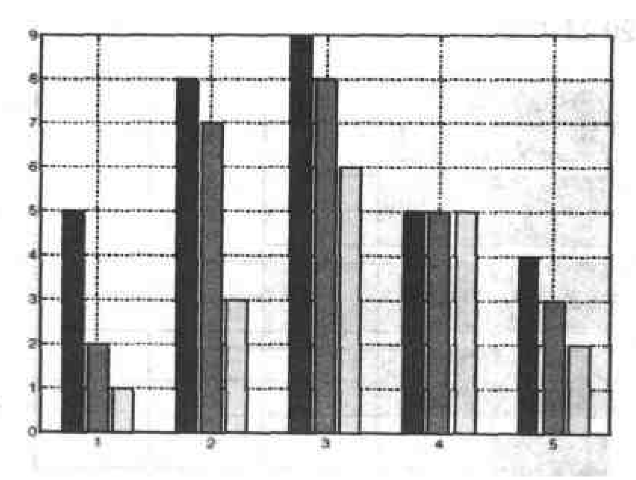


图4-31 简单矩阵的直方图绘制结果

对于竖直直方图而言，每个直方条的高度与每行中的元素总和相等，每一段的高度则等于相应的元素值。例如，对于矩阵Y：

Y = [5 1 2；8 3 7；9 6 8；5 5 5；4 2 3]；

使用bar函数的stack选项来创建堆型直方图：

```
bar(Y,'stack')
grid on
set(gca,'Layer','top') % 显示图形顶端的网线
```

以上语句将创建一个如图4-32所示的二维堆型直方图。

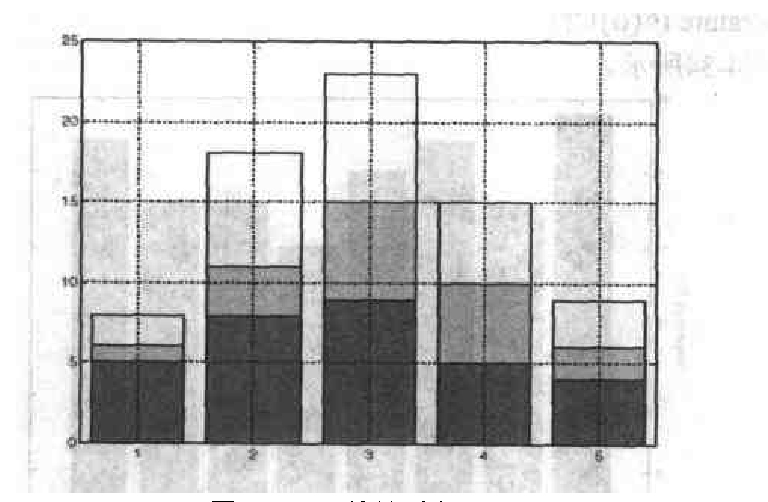


图4-32 二维堆型直方图

对于水平直方图而言，每个直方条的长度等于矩阵每一行的元素总和，每一段等于相应的元素值。例如，使用以下语句绘制上例中Y的水平直方图：

```
barh(Y,'stack')
grid on
set(gca,'Layer','top')
```

绘图结果如图4-33所示。

直方图一般是自动生成x轴及其刻度的，用户也可以通过指定x轴的数值向量（对于水平直方图则指定y向量）来标记坐标轴。例如，给出以下温度数据：

```
temp = [29 23 27 25 20 23 23 27];
```

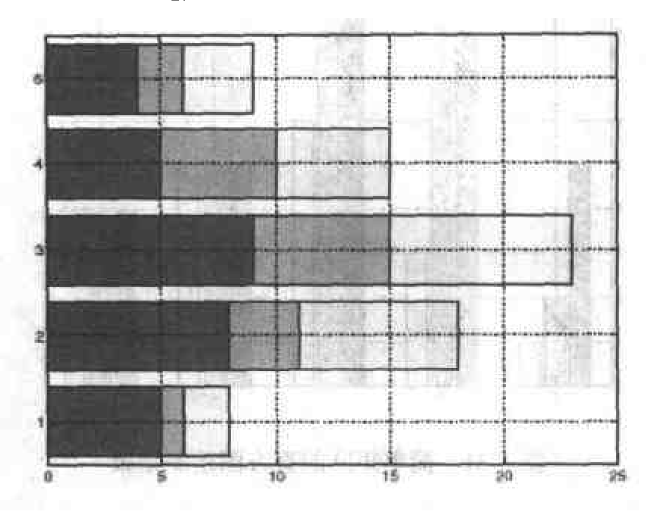


图 4-33　水平直方图

这些温度值是以35天为周期，每隔5天采样得到的：

```
days = 0:5:35;
```

用户使用以下语句显示直方图，其中x轴表示日期，y轴表示当天的测量温度。给各个轴加上标签。

```
bar(days,temp)
xlabel('Day')
ylabel('Temperature (^{o}C)')
```

绘图结果如图4-34所示。

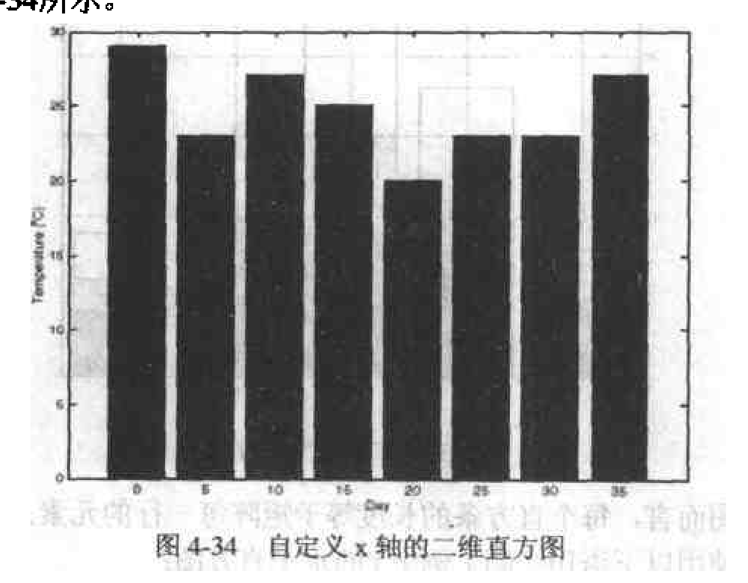


图 4-34　自定义 x 轴的二维直方图

同样可以使用以下语句改变y轴的范围：

```
set(gca,'YLim',[15 30],'Layer','top')
```

绘制结果如图4-35所示。

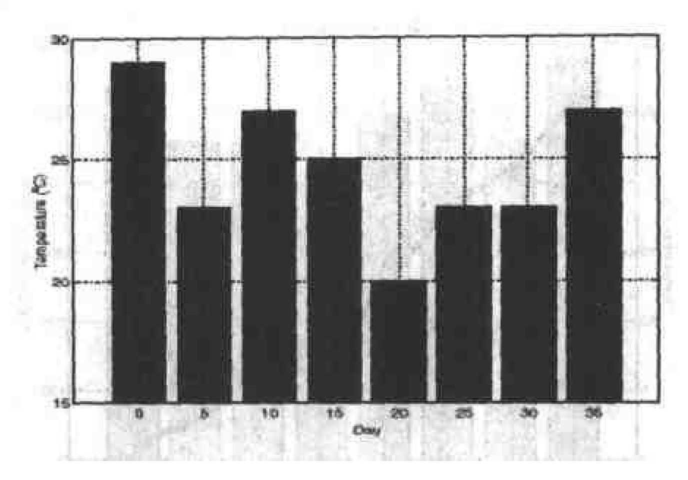


图 4-35 自定义 y 轴的二维直方图

使用在相同位置处创建另一个坐标轴系统的方法，可以实现直方图的数据图形添加工作。例如，考虑一个将有毒废料分解为无毒原料的实验，该实验中的三氯乙烯（TCE）浓度和温度数据为：

```
TCE = [515 420 370 250 135 120 60 20];
temp = [29 23 27 25 20 23 23 27];
```

这些数据是以35天为周期，每隔5天采样获得的：

```
days = 0:5:35;
```

使用以下语句显示直方图并标记坐标轴：

```
bar(days,temp)
xlabel('Day')
ylabel('Temperature (^{o}C)')
```

然后覆盖直方图中的浓度数据，即在与第一个坐标轴相同的位置处再放置一个坐标轴。首先要保存第一个坐标轴的句柄：

```
h1 = gca;
```

在绘制数据前先在与第一个坐标轴相同的位置处创建第二个坐标轴：

```
h2 = axes('Position',get(h1,'Position'));
plot(days,TCE,'LineWidth',3)
```

为了确保第二个坐标轴不会影响第一个坐标轴，将第二个坐标轴的y轴放在右边，并使背景透明，同时不对x轴进行重复标记：

```
set(h2,'YAxisLocation','right','Color','none','XTickLabel',[])
```

重新排列两个坐标轴系统的x轴，并显示直方图顶端的网线：

```
axes set(h2,'XLim',get(h1,'XLim'),'Layer','top')
```

注释图形：

```
text(11,380,'Concentration','Rotation',-55,'FontSize',16)
ylabel('TCE Concentration (PPM)')
title('Bioremediation','FontSize',16)
```

绘制结果如图4-36所示。

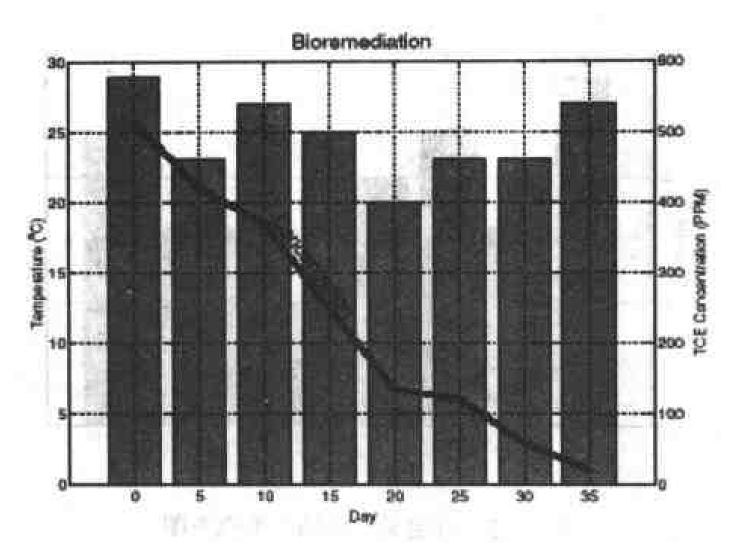


图 4-36 双坐标轴系统的二维直方图

4.3.3 面积图

area 函数显示由向量或多列矩阵生成的多条曲线。area 将矩阵每一列的数值绘制成一条曲线，并将曲线与 x 轴所成的区域填充起来。面积图适合于说明指定 x 位置的向量或矩阵元素在所有元素的总和中所占有的比例。缺省情况下，area 将矩阵每一行的所有元素值加起来，根据这个值来绘制一条曲线。假设有矩阵：

Y = [5 1 2; 8 3 7; 9 6 8; 5 5 5; 4 2 3];

为了显示面积图的网格前景，使用以下语句：

```
area(Y)
  set(gca,'Layer','top')
  set(gca,'XTick',1:5)
```

以上语句将显示一个包含三个面积图的图形（如图 4-37 所示），每一个面积图对应矩阵的一列。面积图的高度是每行元素的总和，每一条连续曲线都以前一条曲线为基准。

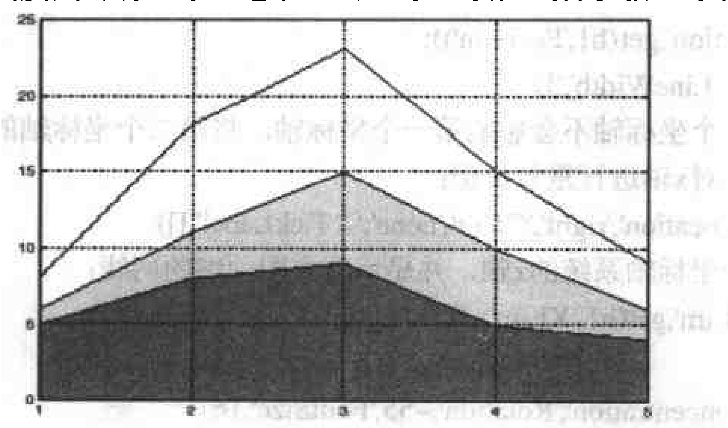


图 4-37 矩阵 Y 的面积图绘制结果

面积图在比较不同的数据集时非常有用。例如：给出一个包含销售指数的向量：

sales = [51.6 82.4 90.8 59.1 47.0];

以上数据以五年为周期：

x = 90:94;

再给出这五年中的利润指数：

profits = [19.3 34.2 61.4 50.5 29.4];

在同一个坐标轴中显示两个不同的面积图：设置面积图的内部颜色（FaceColor）、边缘颜色（EdgeColor）和边缘线宽度(LineWidth)：

```
area(x,sales,'FaceColor',[.5 .9 .6],...
  'EdgeColor','b',...
  'LineWidth',2)
hold on
area(x,profits,'FaceColor',[.9 .85 .7],...
  'EdgeColor','y',...
  'LineWidth',2)
hold off
```

注释图形：

```
set(gca,'XTick',[90:94])
set(gca,'Layer','top')
gtext('\leftarrow Sales')
gtext('Profits')
gtext('Expenses')
xlabel('Years','FontSize',14)
ylabel('Expenses + Profits = Sales in 1,000''s','FontSize',14)
```

绘图结果如图 4-38 所示。

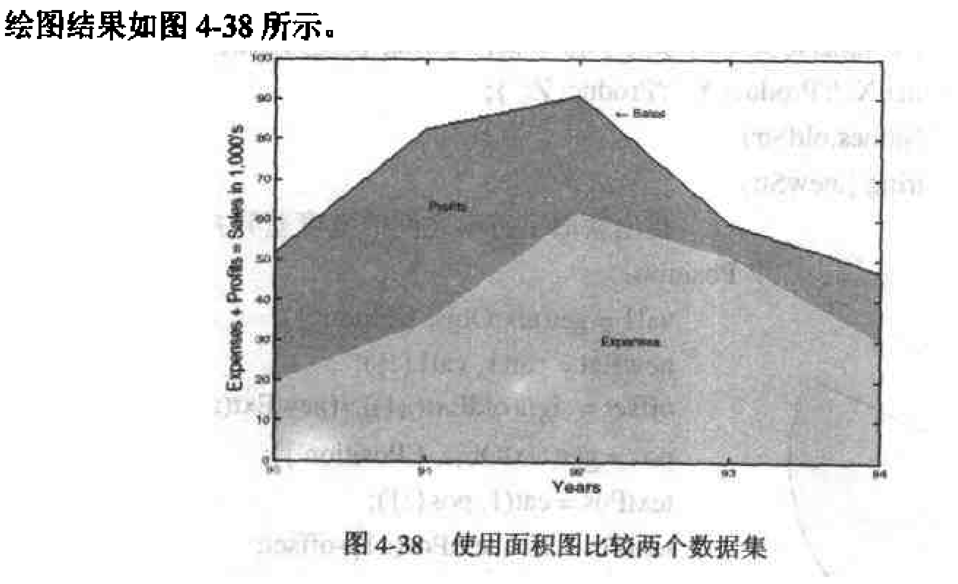


图 4-38 使用面积图比较两个数据集

4.3.4 饼状图表

可以使用 pie 函数创建饼状图表来显示向量或矩阵每一个元素在所有元素总和中所占的比例。下面使用 pie 函数来观察三种产品销售额在所有产品销售额中的比例。给出一个矩阵

X，该矩阵的每一列包含指定一种产品在五年周期内的年销售指数：

```
X = [19.3 22.1 51.6;
     34.2 70.3 82.4;
     61.4 82.9 90.8;
     50.5 54.9 59.1;
     29.4 36.3 47.0];
```

对 X 的每一列进行求和，即计算每一种产品五年中的销售总量：

```
x = sum(X);
```

可以使用 explode 参数来偏移图表中表示最大分布的片段。该参数是一个由非零元素和零元素组成的向量，非零元素表示将从图表中提取的相应片段。首先，创建该向量：

```
explode = zeros(size(x));
```

然后找到分布最多的片段，将相应的 explode 元素设置为 1：

```
[c,offset] = max(x);
explode(offset) = 1;
```

使用以下语句创建被分隔的饼形图表：

```
h = pie(x,explode); colormap summer
```

饼状图表的标签是文本图形对象。为了修改文本字符串的内容和位置，首先要获得该对象的 String 和 Extent 属性：

```
textObjs = findobj(h,'Type','text');
oldStr = get(textObjs,{'String'});
val = get(textObjs,{'Extent'});
oldExt = cat(1,val{:});
```

创建新的字符串，然后将文本对象的字符串属性 String 设置为新的字符串：

```
Names = {'Product X: ';'Product Y: ';'Product Z: '};
newStr = strcat(Names,oldStr);
set(textObjs,{'String'},newStr)
```

找出新旧字符串之间的宽度差异并据此修改位置属性 Position：

```
val1 = get(textObjs, {'Extent'});
newExt = cat(1, val1{:});
offset = sign(oldExt(:,1)).*(newExt(:,3)-oldExt(:,3))/2;
pos = get(textObjs, {'Position'});
textPos = cat(1, pos{:});
textPos(:,1) = textPos(:,1)+offset;
set(textObjs,{'Position'},num2cell(textPos,[3,2]))
```

绘图结果如图 4-39 所示。

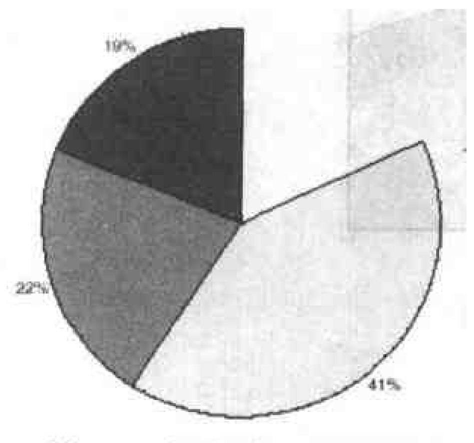


图 4-39　表示矩阵 X 的饼状图表

这里要注意，当 pie 函数的第一个输入参数元素的总和大于或等于 1 时，pie 函数将对数据进行格式化。因此，每个片的面积为 x_i/sum(x_i)，其中 x_i 是 x 的元素。格式化的数据决定每个片所占的比例。如果第一个输入参数元素的总和

小于 1，pie 将不对数据进行格式化，函数将绘制一个不完整的饼。例如：

```
x = [.19 .22 .41];
pie(x)
```

以上语句绘图结果如图 4-40 所示。

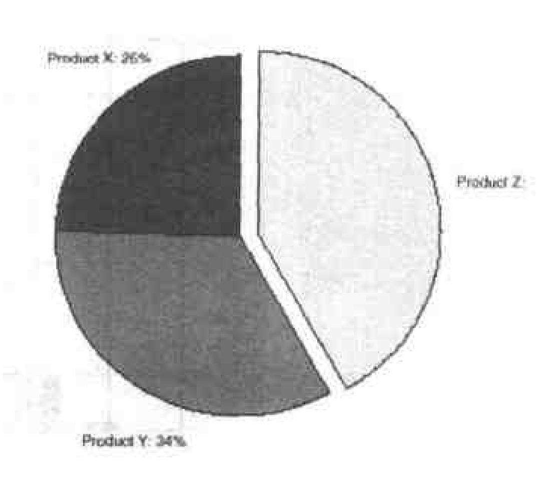


图 4-40 元素和小于 1 的饼状图表

4.3.5 柱状图

MATLAB 柱状图能够显示数据值的分布情况。创建柱状图的函数有两个：hist 和 rose。hist 将在一个笛卡尔坐标系中显示柱状图，而 rose 将在极坐标中显示柱状图。柱状图函数首先要计算某一范围内的元素个数，然后将每个范围显示为一个矩形，矩形的高度表示落在该区间内的元素个数。

hist 函数用一个在矩阵 Y 最大值和最小值之间均匀分布的矩形条来说明 Y 的元素分布情况。如果 Y 是一个向量且 hist 函数仅有 Y 一个输入参数，那么 hist 函数将产生 10 个矩形条，也就是说将整个区间分为 10 个子区间。例如，生成 10000 个随机数据向量 yn，然后绘制 yn 的分布情况：

```
yn = randn(10000,1); hist(yn)
```

结果如图 4-41 所示。

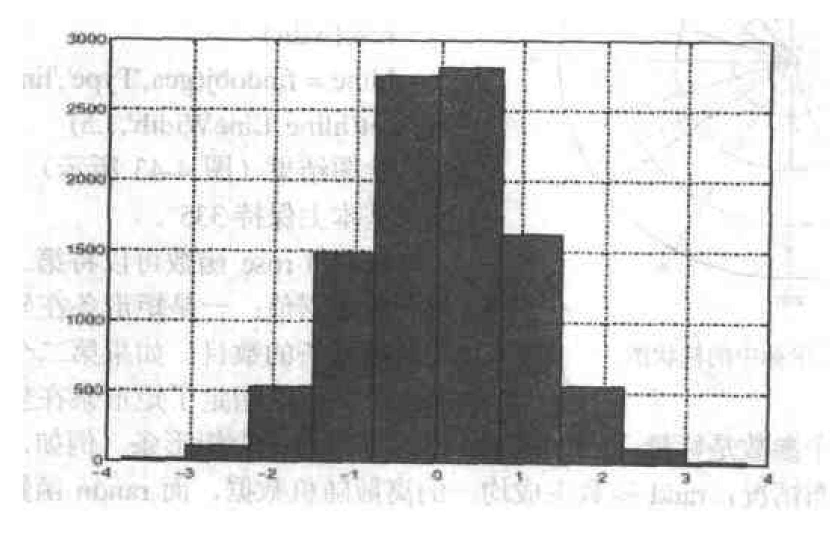


图 4-41 随机向量的柱状图

当 Y 是一个矩阵时，hist 为矩阵的每一列创建一个矩形条集合，并使用不同的颜色来显示不同的集合。

例如，以下语句：

```
Y = randn(10000,3);
hist(Y)
```

将产生如图 4-42 所示的绘图结果。

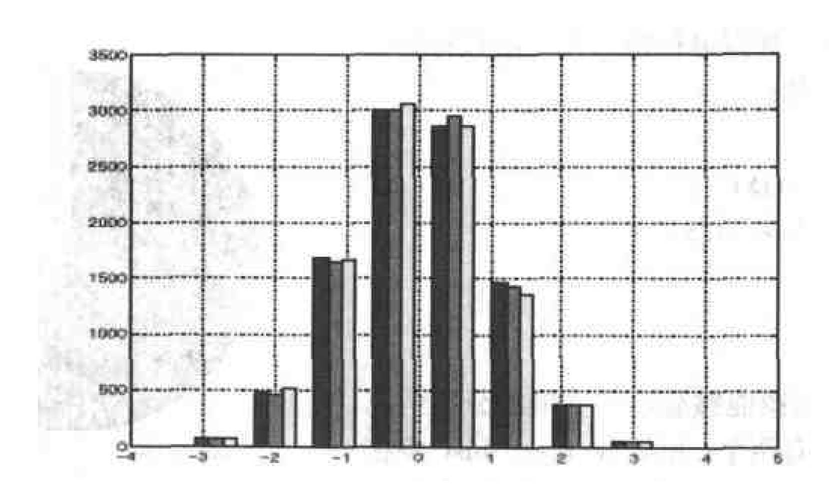


图 4-42　随机矩阵的柱状图

下面介绍极坐标轴的柱状图绘制方法。考虑一个描述 12 小时内风向情况的例子。风向数据为：

```
wdir = [45 90 90 45 360 335 360 270 335 270 335 335];
```

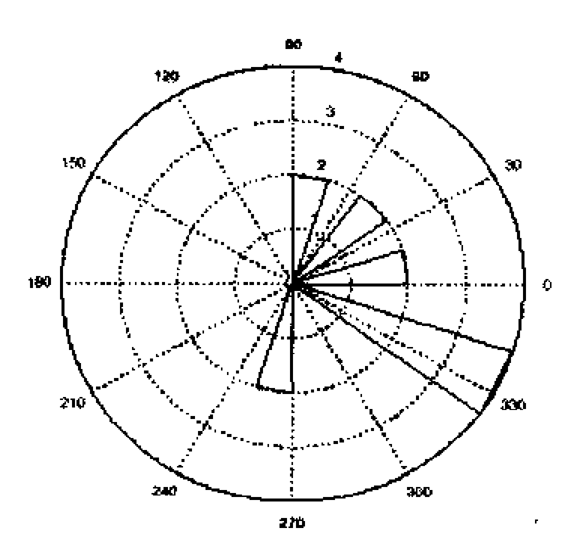


图 4-43　极坐标中的柱状图

为了使用 rose 函数显示这些数据，首先要将这些角度数据转换为弧度，然后使用弧度数据作为 rose 函数的参数。为了清楚地表明风向，还需要增加输出线条的宽度：

```
wdir = wdir * pi/180;
rose(wdir)
hline = findobj(gca,'Type','line');
set(hline,'LineWidth',1.5)
```

绘图结果（图 4-43 所示）说明 12 小时内的风向基本上保持 335°。

hist 和 rose 函数可以将第二个输入参数理解为两种数据值：一是矩形条在坐标轴中的位置，二是矩形条的数目。如果第二个参数是向量 x，那么该参数就指定了矩形条在坐标轴中的位置和分布；如果第二个参数是标量，hist 和 rose 将元素分为 x 个矩形条。例如，比较两个随机函数生成数据的分布情况：rand 函数生成均一的离散随机数据，而 randn 函数则产生普通的离散随机数据。

```
yn = randn(10000,1);
yu = rand(10000,1);
```

第一个柱状图显示了由 randn 函数生成的随机数据的分布，矩形条在 x 轴中的位置和数目由向量 x 决定：

```
x = min(yn):.2:max(yn);
subplot(1,2,1)
hist(yn,x)
title('Normally Distributed Random Numbers','FontSize',16)
```

第二个柱状图显示 rand 函数生成的随机数据分布，图形将精确地生成 25 个矩形条：

```
subplot(1,2,2)
hist(yu,25)
title('Uniformly Distributed Random Numbers','FontSize',16)
```

以上语句绘制的两个柱状图见图 4-44 所示。

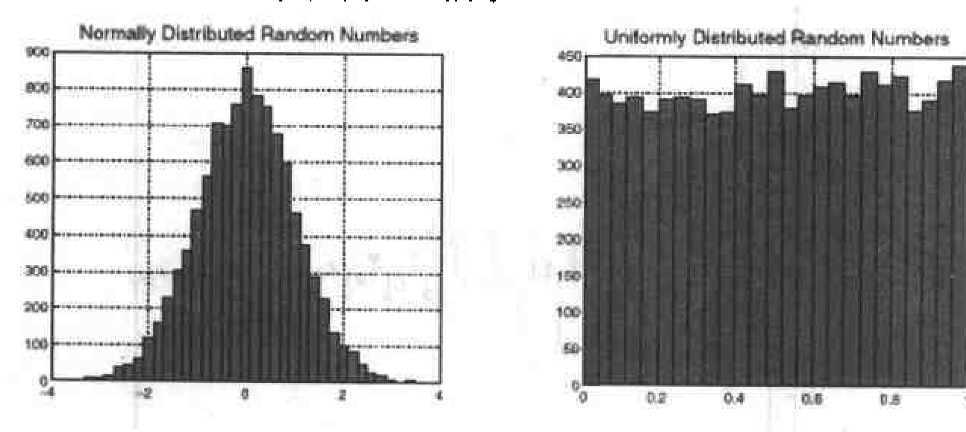


图 4-44 指定矩形条数目的柱状图

4.3.6 枝干图和阶梯图

枝干图将绘制一系列在数据值处以标记符号终止的直线来描述数据。二维枝干图是沿 x 轴扩展的。可以用 stem 函数显示二维离散序列数据，例如，计算函数：

$$y = e^{-\alpha t}\cos\beta t$$

在以下取值情况下的函数值：

```
alpha = .02; beta = .5; t = 0:4:200;
y = exp(-alpha*t).*sin(beta*t);
```

以上语句将产生离散向量 y，使用 plot 函数绘制 y 将产生图 4-45 所示的结果，每个数据点都被一条直线连接起来。

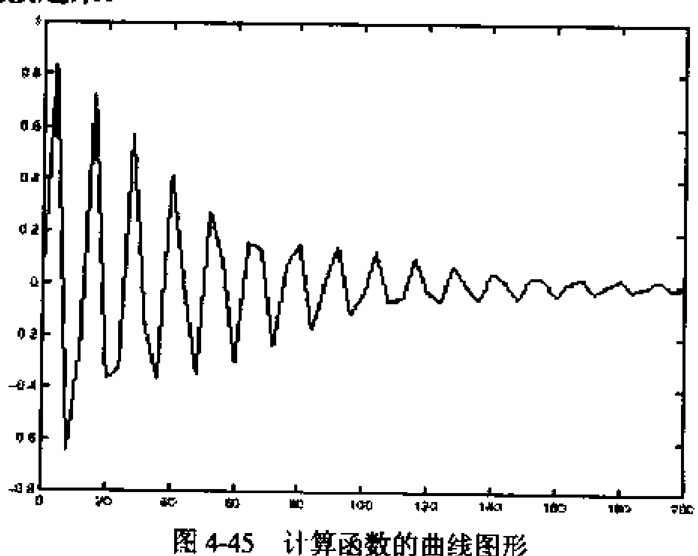


图 4-45 计算函数的曲线图形

下面绘制数据 y 的枝干图。为枝干图的坐标轴添加标签：

```
xlabel('Time in \musecs')
ylabel('Magnitude')
```

绘制结果如图 4-46 所示。

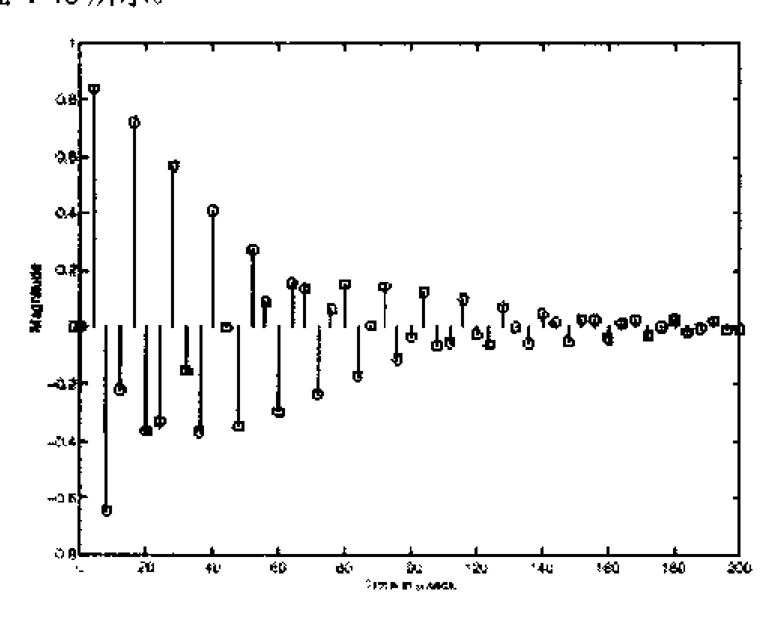


图 4-46　计算函数的枝干图形

如果用户仅使用一个输入参数，那么采样（枝干）的数目将与参数长度相等。在本例中，采样数目是 t 的函数，t 包括 51 个元素。

用户可以指定枝干图的线型、标示符的种类以及枝干图所用的颜色。例如，字符串'sr'表示指定一条点线、一个正方形标记符和红色枝干图。fill 参数表示将要填充标记符：

```
stem(t,y,'- -sr','fill')
```

绘图结果如图 4-47 所示。

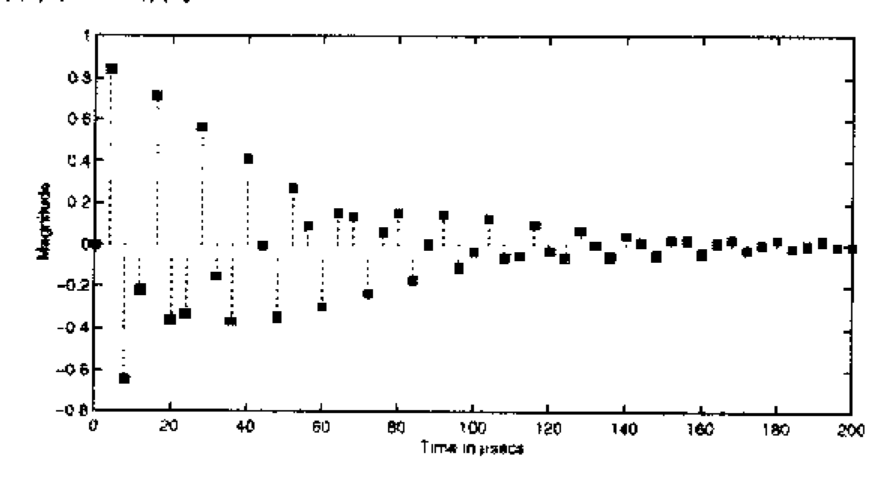


图 4-47　用户定制的枝干图

有时联合使用曲线图和枝干图能够帮助用户理解推导结论的过程。例如，创建一个包含 60 个元素的向量并定义两个函数 a 和 b：

```
x = linspace(0,2*pi,60);
```

```
a = sin(x);
b = cos(x);
```

创建枝干图来显示这两个函数的叠加效果：

```
stem_handles = stem(x,a+b);
```

为了更好地理解该叠加效果，绘制 a 和 b 曲线。在绘制曲线之前，要保证以前的枝干图不被擦掉：

```
hold on
plot_handles = plot(x,a,'- -r',x,b,'- -g');
hold off
```

使用图例来标注图形。这里要注意，stem 函数包含两种线条对象，一是标示符的线条，二是垂直线条。这里使用 stem 返回的第一个句柄（表示标示符线条）来创建图例：

```
legend_handles = [stem_handles(1);plot_handles];
legend(legend_handles,'a + b','a = sin(x)','b = cos(x)')
```

添加坐标轴标签和图形标题：

```
xlabel('Time in \musecs')
ylabel('Magnitude')
title('Linear Combination of Two Functions')
```

绘图结果如图 4-48 所示。

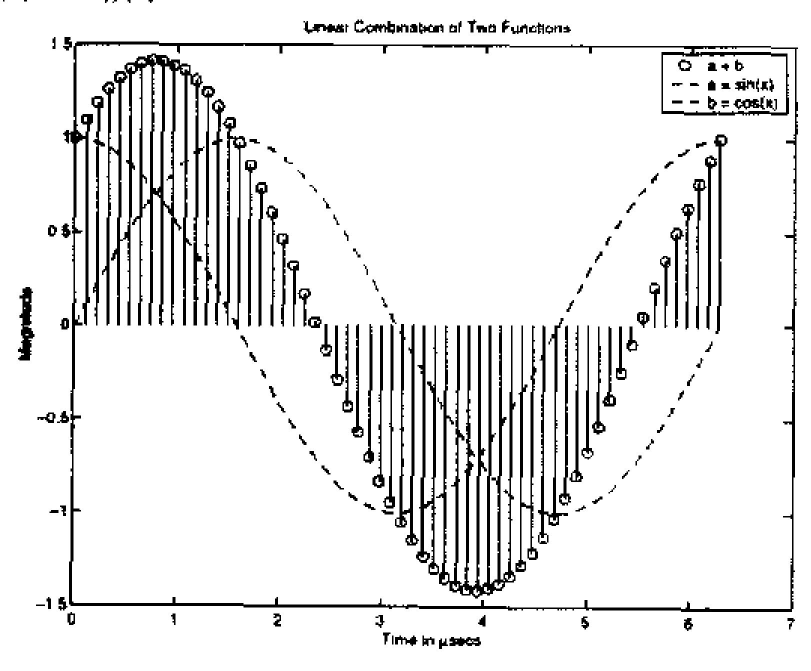


图 4-48　曲线图和枝干图的联合使用

4.3.7　阶梯图

阶梯图将数据显示为固定间隔的边缘图形，也就是说，这种图形将保持 x(i)（i 是 x 数据的索引）和 x(i+1)之间所有数据的 y 值不变。这种图形适合于显示数字采样系统的记录。下面给出一个绘制函数阶梯图的例子。

首先定义一个时变函数：

```
alpha = 0.01;
beta = 0.5;
t = 0:10;
f = exp(-alpha*t).*sin(beta*t);
```

使用 stairs 将该函数显示为阶梯图形并绘制一条插值曲线：

```
stairs(t,f)
hold on
plot(t,f,'--*')
hold off
```

注释图形并设置坐标轴范围：

```
label = 'Stairstep plot of e^{-(\alpha*t)} sin\beta*t';
text(0.5,-0.2,label,'FontSize',14)
xlabel('t = 0:10','FontSize',14)
axis([0 10 -1.2 1.2])
```

绘制结果如图 4-49 所示。

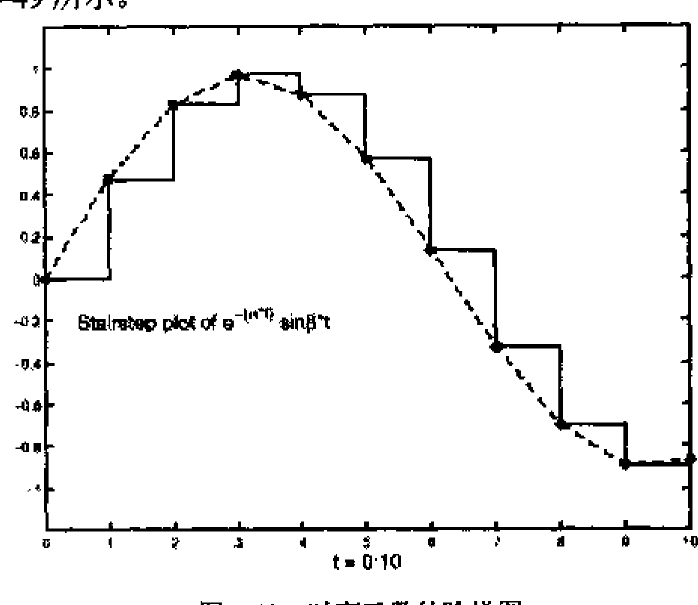


图 4-49　时变函数的阶梯图

4.3.8　方向和速率图形

有三个函数可以显示同时具有方向和速度向量的数据的二维图形：compass、feather 和 quiver。用户可以使用一个或两个参数来定义向量，如果使用两个参数，第一个参数表示向量的 x 分量，第二个表示 y 分量。如果只定义一个参数，则函数将该参数视为复数，实部表示 x 分量，虚部表示 y 分量。

1. 罗盘图形（compass）

compass 函数显示从原点向外发散的向量。函数使用笛卡尔坐标系和圆形网格线。下例的罗盘图形表明了 12 小时风向和强度信息。

风向和强度数据定义如下：

```
wdir = [45 90 90 45 360 335 360 270 335 270 335 335];
knots = [6 6 8 6 3 9 6 8 9 10 14 12];
```

在将风向坐标转换为笛卡尔坐标之前，首先将风向的角度值转换为弧度值：

```
rdir = wdir * pi/180;
[x,y] = pol2cart(rdir,knots);
compass(x,y)
```

创建注释文本：

```
desc = {'Wind Direction and Strength at',
'Logan Airport for ',
'Nov. 3 at 1800 through',
'Nov. 4 at 0600'};
text(-28,15,desc)
```

绘图结果如图 4-50 所示。

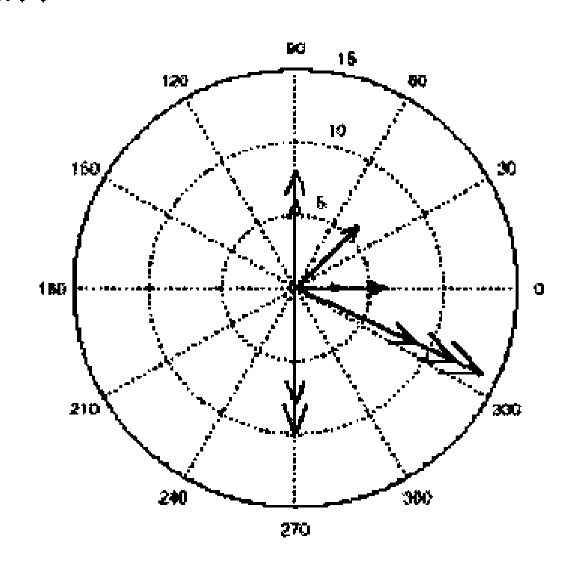


图 4-50 风向的罗盘图形

2. 羽状图（feather）

feather 函数表明向量沿 x 轴方向发散的情况。例如，创建一个角度从 90o到 0o变化的向量和一个大小与该向量相同、元素值为 1 的向量：

```
theta = 90:-10:0;
r = ones(size(theta));
```

创建羽状图之前，将数据坐标转换为笛卡尔坐标，同时增大 r 的幅值使箭头更为明显：

```
[u,v] = pol2cart(theta*pi/180,r*10);
feather(u,v)
axis equal
```

绘制结果如图 4-51 所示。

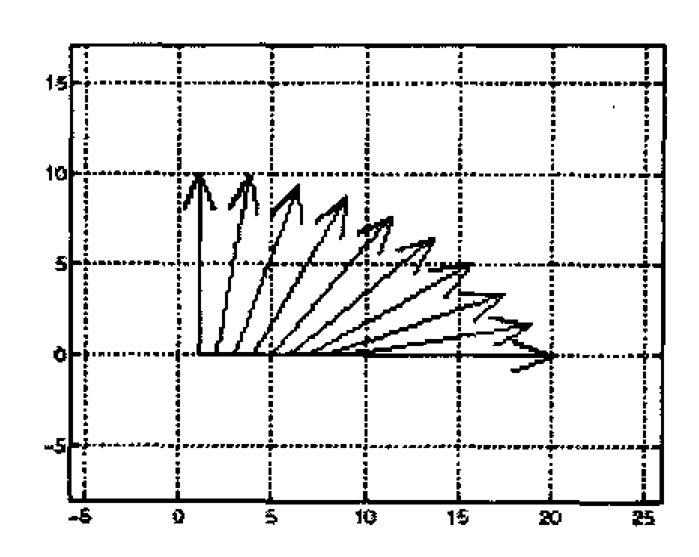


图 4-51　变化向量的羽状图

如果输入参数 Z 是一个复数矩阵，feather 函数将 Z 的实部作为向量的 x 分量，虚部作为向量的 y 分量：

```
t = 0:0.5:10; % 时间范围
s = 0.05+i; % 螺线率
Z = exp(-s*t); % 计算衰减指数
feather(Z)
```

绘制结果如图 4-52 所示。

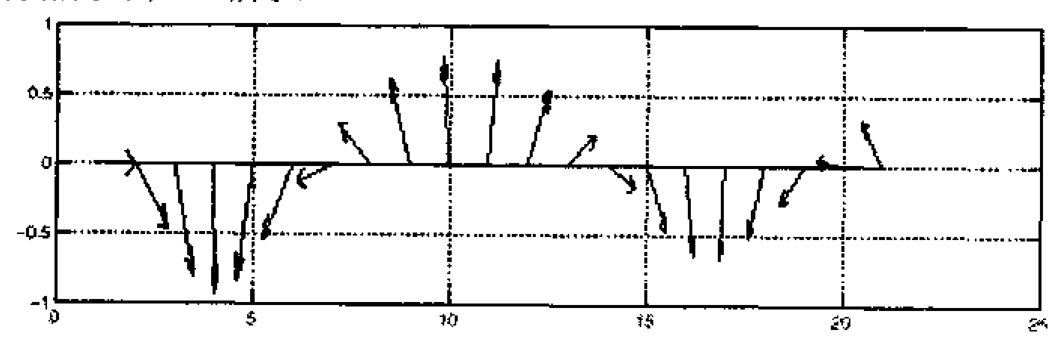


图 4-52　复数矩阵的羽状图

3. 二维箭头图形（quiver）

quiver 函数在二维空间指定点处显示向量。箭头图形一般都是与其他图形配合使用的，例如，创建 peaks 函数的 10 条等高线：

```
n = -2.0:.2:2.0;
[X,Y,Z] = peaks(n);
contour(X,Y,Z,10)
```

绘制结果如图 4-53 所示。

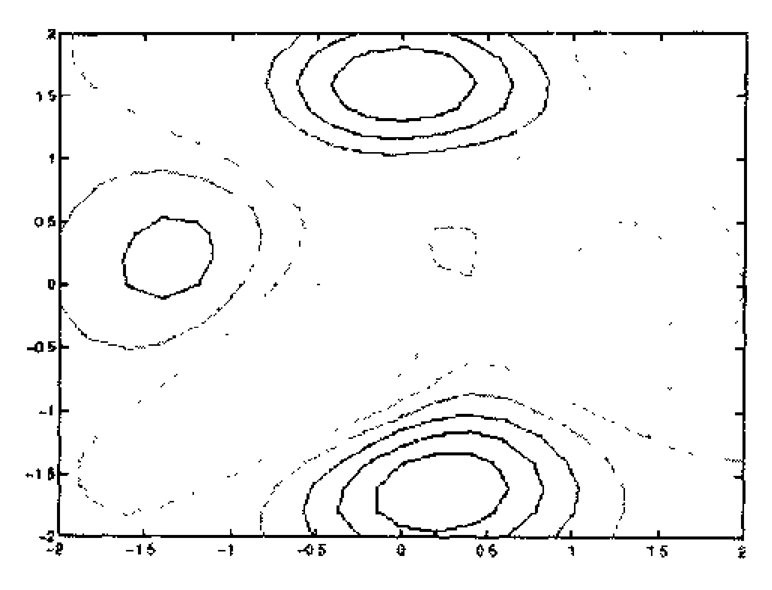


图 4-53 peaks 函数的等值线图

现在使用 gradient 函数来创建 quiver 函数的输入向量：

[U,V] = gradient(Z,.2);

然后绘制箭头图形：

hold on

quiver(X,Y,U,V)

hold off

绘制结果如图 4-54 所示。

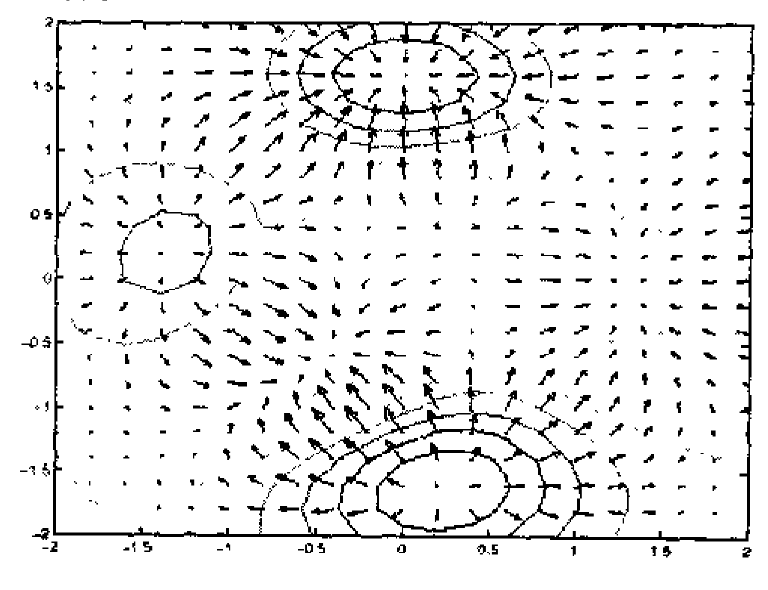


图 4-54 peaks 函数的箭头图形

4.3.9 等高线图

等高线图将创建、显示并标注由一个或多个矩阵决定的等值线。

等高线绘制函数 sontour 仅有一个输入参数是必需的：平面高度矩阵。此时函数将根据

数据的最大值和最小值来决定等值线的条数。如果希望明确定义所需的等值线条数，可以使用可选的第二个参数来实现。假设需要绘制 peaks 函数的等值线图形：

```
[X,Y,Z] = peaks;
contour(X,Y,Z,20)
```

将会生成一幅包含 20 条等值线的二维图形（如图 4-55 所示）。

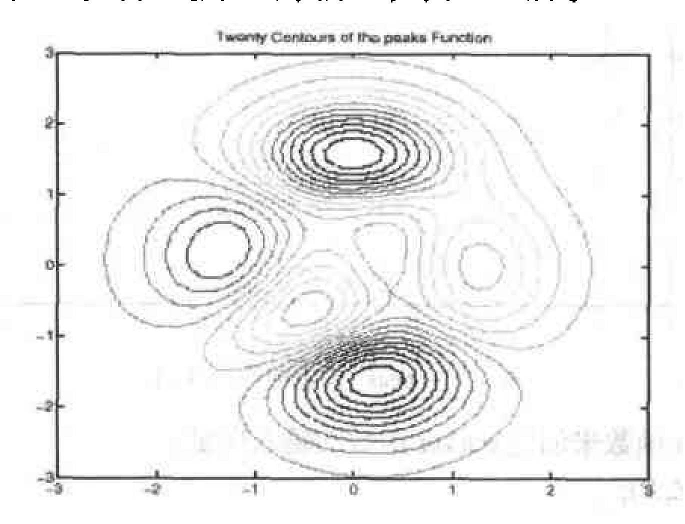


图 4-55　包含 20 条曲线线的等值线图

每一条等值线都有一个关联数值，clabel 函数使用这个数值作为二维等值线的显示标签。等值线矩阵将包含这个关联数值，例如显示 peaks 函数的 10 条等值线：

```
Z = peaks;
[C,h] = contour(Z,10);
```

然后给等值线添加标签并显示标题：

```
clabel(C,h)
title({'Contour Labeled Using','clabel(C,h)'})
```

注意 clabel 函数只为那些间隔足够插入内置标签的等值线添加标签。绘制结果如图 4-56 所示。

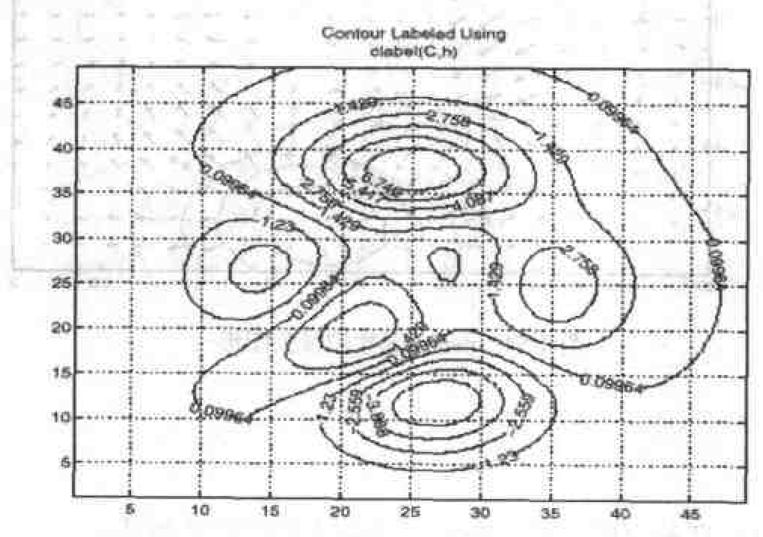


图 4-56　带标签的等值线图

contourf 函数用来显示二维等值线图形并填充等值线中间的区域。使用 caxis 来控制等值线与颜色的映射。例如，以下语句将等值线填充颜色映射到调色板的中心位置：

```
Z = peaks;
[C,h] = contourf(Z,10);
caxis([20 20])
title({'Filled Contour Plot Using','contourf(Z,10)'})
```

绘图结果如图 4-57 所示。

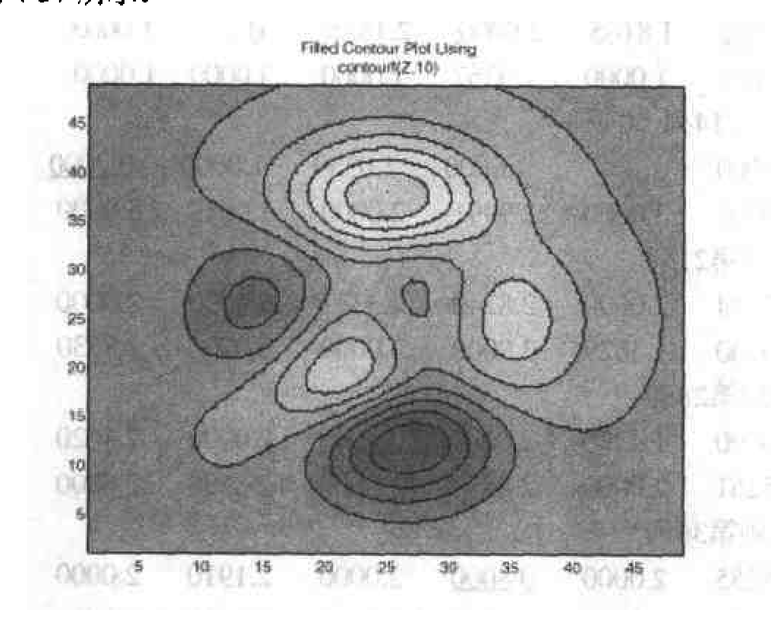


图 4-57 填充性等值线图

等值线函数允许用户在函数的第二个参数中指定等值线分级数目或绘制特殊水平（即关联数值）的等值线。如果第二个参数是向量，则指定待绘制等值线的水平，如果是个标量则指定等值线的数目。MATLAB 不区分单元素向量和标量，因此，如果函数的第二个参数是单元素向量，那么函数就将其理解为等值线的数目。

为了显示指定水平的等值线，将函数的第二个参数 v 定义为一个两元素向量，每一个元素都等于所需的等值线水平。例如，创建 peaks 函数的三维等值线：

```
xrange = -3:.125:3;
yrange = xrange;
[X,Y] = meshgrid(xrange,yrange);
Z = peaks(X,Y);
contour3(X,Y,Z)
```

为了仅显示 Z=1 等值线，定义 v 为[1 1]：

```
v = [1 1]
contour3(X,Y,Z,v)
```

contourc 函数为其他等值线函数计算等值矩阵，这是一个不能被命令行调用的低级命令。等值算法首先确定哪个水平的等值线将被绘制：如果用户指定输入参数 v，v 的元素代表等值线水平，v 的长度决定产生的等值线的条数；如果用户不指定 v，算法选择不超过 20 个（数目能够被 2 或 5 整除）等值水平。等值算法将输入矩阵 Z 视为一个有规律的空间网

格，Z 的每一个元素都与其邻近元素相连。算法通过比较每个由四个邻近点组成的矩阵单元和等值水平来扫描矩阵，如果等值水平落在某一个单元中，算法将使用线性插值方法将点放在等值线经过的单元上。算法将这些点连接起来成为等值线的一段。

contour、contour3 和 contourf 返回一个指定所有等值线的行数为 2 的矩阵。下面举例来说明该矩阵的格式：

假设等值矩阵由语句 C = contour(peaks(3))生成，则矩阵数据为：

```
第 1 列到第 7 列
    -2.000    1.8165    2.0000    2.1835    0         1.0003    2.0000
    3.0000    1.0000    1.0367    1.0000    3.0000    1.0000    1.1998
第8列到第14列
    3.0000    0         1.0000    1.0359    1.0000    0.2000    1.6669
    1.0002    3.0000    2.9991    2.0000    1.0018    5.0000    3.0000
第15列到第21列
    1.2324    2.0000    2.8240    2.3331    0.4000    2.0000    2.6130
    2.0000    1.3629    2.0000    3.0000    5.0000    2.8530    2.0000
第22列到第28列
    2.0000    1 4290    2.0000    0.6000    2.0000    2.4020    2.0000
    1.5261    2.0000    2.8530    5.0000    2.5594    2.0000    1.6892
第29列到第36列
    1.6255    2.0000    0.8000    2.0000    2.1910    2.0000    1.8221    2.0000
    2.0000    2.5594    5 0000    2.2657    2.0000    1.8524    2.0000    2.2657
```

第一行中标有下划线的数值表示等值线水平，该数值所对应的列的第二行元素表示各等值线上点的个数。例如，第一行第 13 列的数值表示水平 0.2000，第二行第 13 列的数值说明该水平的等值线包括 5 个点，这 5 个点的 x 坐标值分别由第一行第 14 列到第 18 列的数值表示，y 坐标值由第二行第 14 列到第 18 列的数值表示。

4.3.10　交互式绘图

由于 ginput 函数能够返回鼠标点的位置，所以用户可以使用鼠标或箭头键来选择绘图点。本例说明了如何使用 ginput 和 spline 函数来创建一个使用二维插值方法获得的曲线。

首先选择平面内的点列[x，y]，然后以步长 0.1 对这些点进行二维插值计算：

```
axis([0 10 0 10])
hold on
    xy = [ ];      % 开始时点列为空
n = 0;
disp('Left mouse button picks points.')    %循环获得点的坐标
disp('Right mouse button picks last point.')
but = 1;
while but = = 1
[xi,yi,but] = ginput(1);
```

```
plot(xi,yi,'ro')
n = n+1;
xy(:,n) = [xi;yi];
end
t = 1:n;    %使用弧线和更好的间隔修改输出
ts = 1: 0.1: n;
xys = spline(t,xy,ts);
plot(xys(1,:),xys(2,:),'b-');    % 绘制修改后的曲线
hold off
```

图4-58是一个输出示例。

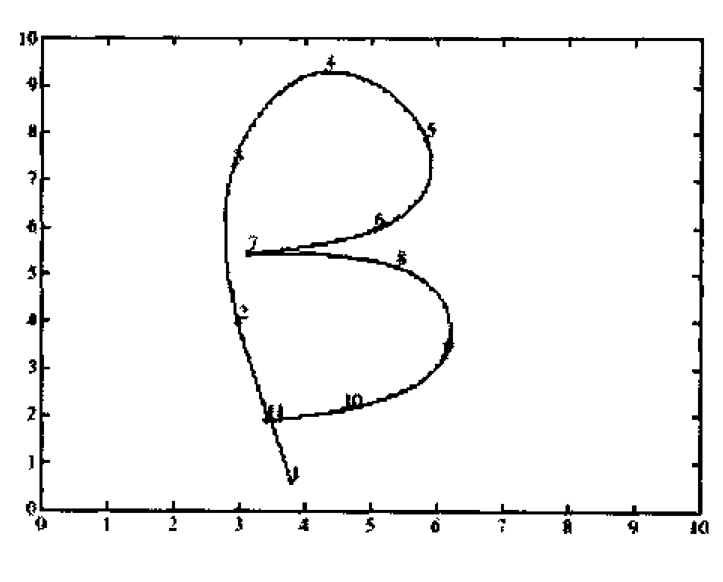


图 4-58　交互式图形输出范例

4.4 实例讲解

【实例一】：人们已经意识到太阳黑子活动是周期性的，大约每11年达到一个高潮。假设已获得了近300年来太阳黑子数量和大小的数据文件sunspot.dat，如何确认这一情况并将该信息用图形尽可能清楚地表达出来？

分析：要实现一项具体的数据绘制任务，最好按照第三章第一节中介绍的绘图步骤来完成。在本问题中，如果要分析太阳黑子的活动周期，就要利用快速傅立叶变换来进行（原理参见有关信号处理书籍）。

解决方法：

步骤一：数据准备。首先将数据文件调入本程序中：

```
load sunspot.dat
```

然后进行数据分析：

```
wolfer=sunspot (:,2);
```

```
Y=fft(wolfer);
 n=length(Y);
Y[1]=[ ];
power=abs(Y(1:n/2)),^2;
nyquist=1/2;
freq=(1:n/2)/(n/2)*nyquist;
period=1./freq;
```

步骤二：选择窗口和图形在窗口中的位置。本例将创建一个新的图形窗口并在默认位置处输出。

步骤三：调用绘图函数

```
plot(period,power)
```

步骤四：设置图形对象的属性。本例使用默认线型和颜色。

步骤五：设置其他能够增强图形可视性的属性：

```
axis([0 40 0 2e+7]);
```

步骤六：标注图形以增强图形的可读性：

```
title(''Fourier Coefficients in the Complex Planc'');
xlabel(''Real Axis'');
ylabel(''Imaginary Axis'');
```

步骤七：输出图形：如图 4-59 所示。显然可以看出，太阳黑子的活动周期大约为 11 年。

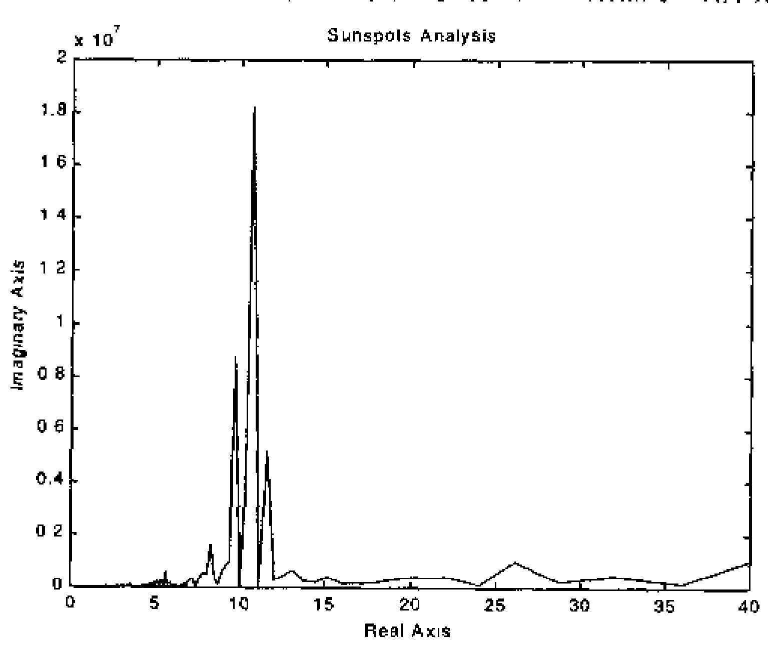


图 4-59　太阳黑子活动分析

【实例二】：给出一幅图像 rice.tif（如图 4-60 所示），可以看出，该图像中心处的背景要比别处亮，如何对该图像进行处理得到背景亮度一致的同一幅图像？

分析：事实上，本例是为了向读者介绍 MATLAB 图像处理工具箱的使用方法。通过使用该工具箱中的图像处理函数可以非常简单地获得处理后的图像。

解决方法：

步骤一：首先装载图像文件：

```
rice=imread('rice.tif');
```

步骤二：将图像数据转换为双精度类型：

```
rice=double(rice)/255;
```

步骤三：调用函数 blkproc，该函数通过对图像的每个 32×32 小块进行扫描找到最小亮度来粗略估计整幅图像的亮度，然后 imresize 将估计得到的亮度扩展为与原图像相同大小。bicubic 插值法保证了数据的平滑性。最后在图像中将计算所得的背景亮度减掉（这导致了物体亮度同时衰减）并使用 imadjust 调节像素亮度使之符合整个亮度范围：

```
bg32=blkproc(rice,[32 32],'min(x(:))');
bg256=imresize(bg32,[256 256],'bicubic');
d=rice-bg256;
adjusted=imadjust(d,[0 max(d(:))],[0 1],1);
```

显示处理后的结果（如图 4-61 所示）：

```
imshow(adjusted)
```

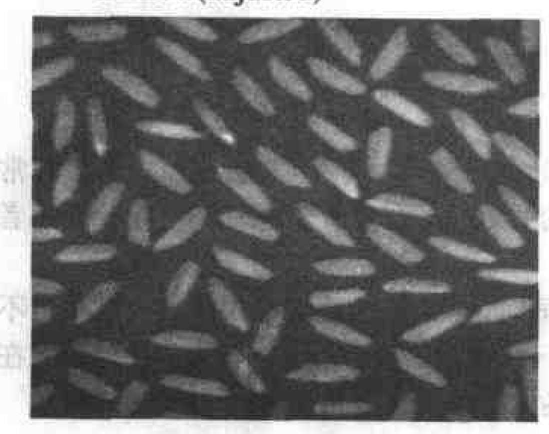


图 4-60 待处理的亮度不均图像

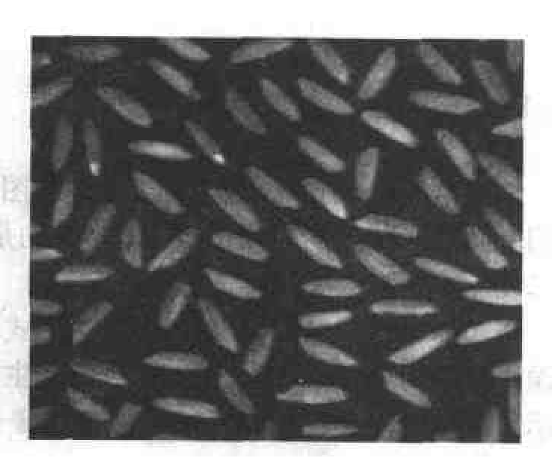


图 4-61 处理后背景亮度均一的图像

4.5 小 结

在本章中读者可以通过大量实例掌握 MATLAB 二维基本图形和特殊图形的绘制方法、图形外观的设置方法以及图像显示和操作方法。读者可能已经感到，在绘制数据时，二维图形的可读性是有限的，能够获得的信息量也是有限的。为此，以后的章节主要向读者介绍三维图形的有关知识。

第 5 章　MATLAB 三维图形

现实生活，尤其是科学计算及工程应用领域中的很多问题都可以抽象为三维空间的问题。前一章中介绍的二维图形虽然也能够有效地表达数据，获得较为直观的信息，但是对某些问题而言，信息量还是不够的。在本章中，读者将学习怎样根据数据绘制的三维曲线或曲面图形以及如何为图形添加必要的说明注释。本章还针对特殊性质的数据介绍了几种特殊三维图形函数表示数据的方法。

5.1　三维曲线图形

5.1.1　三维曲线基本绘图命令

使用绘图函数 plot3 来绘制三维曲线图形。plot3 函数的使用方法与 plot 函数非常类似，它们之间的主要区别主要在于前者输出的是三维图形，含有 z 坐标轴的分量，而后者输出的是二维图形，只有 x 和 y 分量。

与 plot 函数一样，当输入参数的形式不同时，plot3 函数的形式和输出结果也不同。如果输入参数为向量(x,y,z)，则 plot3(x,y,z)生成一条通过各个 (x,y,z) 点的曲线，并且在屏幕上显示它的二维投影。例如，以下语句生成一条如图 5-1 所示的螺线。

```
t = 0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
axis square;
grid on
```

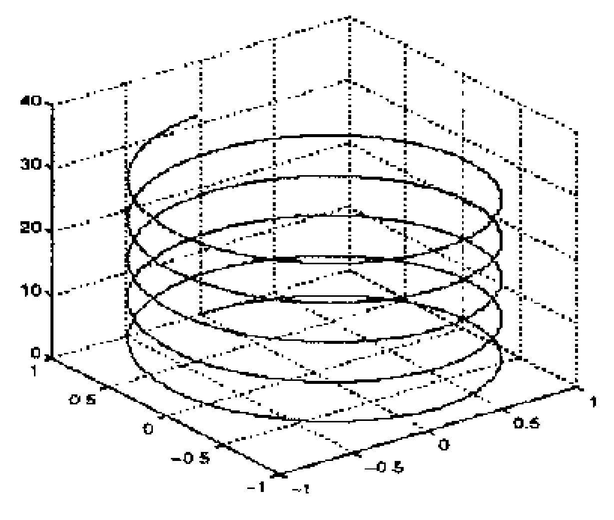


图 5-1　三维向量曲线图形

从图中可以看出，三维曲线图形的坐标轴设置方法与二维图形相同，产生的效果也类似。事实上，plot3 函数与 plot 函数参数选项的含义也是类似的。如果希望设置图形中线条的线型、颜色及标记符，可以使用上一章中介绍的各种特征字符串来实现。

如果输入参数是三个维数相同的矩阵 X，Y 和 Z，plot3(X,Y,Z)将绘制 X，Y，Z 每一列的数据曲线。例如：

```
[X,Y] = meshgrid([-2:0.1:2]);
Z = X.*exp(-X.^2-Y.^2);
plot3(X,Y,Z)
grid on
```

绘图结果如图 5-2 所示。

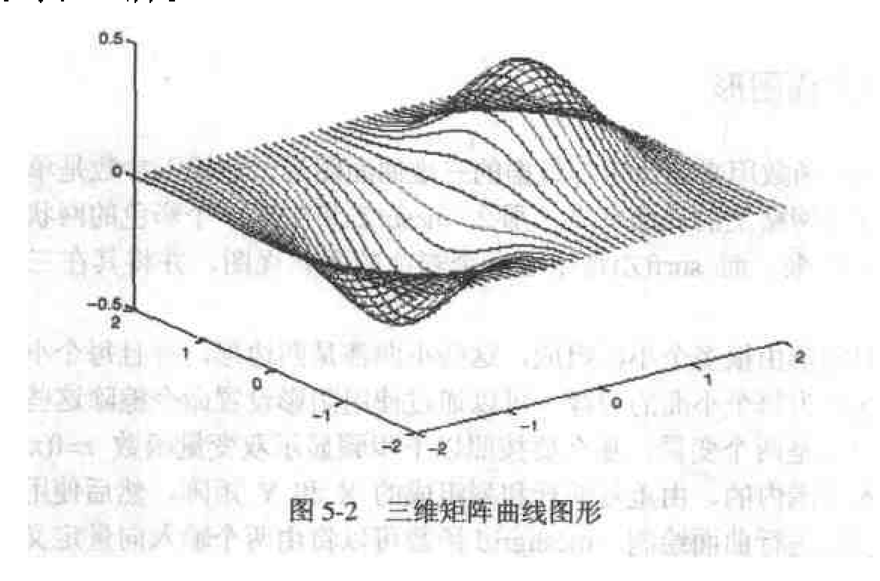


图 5-2　三维矩阵曲线图形

5.1.2　三维图形的坐标轴标签和图形标题

与二维图形相比，三维图形多了一个 z 轴，因而对坐标轴的标记也要多进行一次，也就是说，在对三维坐标轴进行标记时要使用三个函数：xlabel、ylabel 和 zlabel。

zlabel 函数的使用方法与上一章中介绍的 xlabel 和 ylabel 函数类似。可以使用以下语句给图 5-2 所示的三维曲线图形加上标题和坐标轴标签：

```
xlabel('x')
ylabel('Y')
zlabel(' X.*exp(-X.^2-Y.^2)')
```

5.2　三维曲面图形

5.2.1　三维曲面图形介绍

MATLAB 用 x，y 平面内矩形网格中的点的 z 坐标来定义曲面，曲面图形由连接相邻点的曲线组成。当矩阵过大用数字形式难以表示，或使用双变量的图形函数时，绘制曲面图形

将十分有用。

MATLAB 可以生成两种形式的曲面图：网格图是一种仅对连接曲线着色的曲面图，而曲面图对连接线及连接线构成的表面都进行着色。

常用的三维图形函数主要有：

mesh，surf：曲面绘制；

meshc，surfc：绘制曲面的等高线；

meshz：绘制曲面的参考平面；

pcolor：绘制平面；

surfl：绘制设置指定方向光源的曲面；

surface：创建曲面图形对象的底层函数。

5.2.2　网格和曲面图形

mesh 和 surf 函数用来创建矩阵数据的三维曲面图形。若输入参数是单独的矩阵 Z，其元素 Z(i,j)定义(i,j)网格上的曲面高度，那么 mesh(Z)将生成一个彩色的网状视图，并将其在三维坐标中显示出来。而 surf(Z)将生成一个彩色的面状视图，并将其在三维坐标中显示出来。

通常，曲面图形由很多个小面组成，这些小面都是四边形，并且每个小面都有一定的颜色，黑色的网线作为每个小面的边缘。可以通过使用阴影设置命令擦除这些网线。

如果输入参数是两个变量，那么要按照以下步骤显示双变量函数 z=f(x,y)的曲面图形：首先要生成输入范围内的、由重复的行和列组成的 X 和 Y 矩阵，然后使用这两个矩阵对函数进行求值，最后进行曲面绘制。meshgrid 函数可以将由两个输入向量定义的函数区域转换为两个矩阵 X 和 Y，矩阵 X 的行向量与向量 x 相同，矩阵 Y 的列向量与向量 y 相同。

为了说明 meshgrid 函数的使用方法，考虑表达式：

sinc=sin(r)/r

为了计算该函数在-8≤x≤8，-8≤y≤8 范围内、间隔为 0.5 的点的取值，用户需要将这个均匀的数据范围参数传递给 meshgrid，该函数将在 x 和 y 两个方向上使用这个参数：

```
[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2) + eps;
```

上式中矩阵 R 代表原点到矩阵中心的距离。使用 eps 防止零作为除数从而产生无穷结果。使用 sinc 函数并使用 mesh 绘制 Z 的三维曲面：

```
Z = sin(R)./R;
mesh(Z)
```

绘制结果如图 5-3 所示。

在上例中使用的函数空间是一个均匀的网格区间，事实上用户也可以使用 meshgrid 来创建简单的非均匀数据空间上的网格，然后进行计算和绘图。网格生成后 MATLAB 将相邻的矩阵元素连接成四边形网格，构造所需的曲面图形。

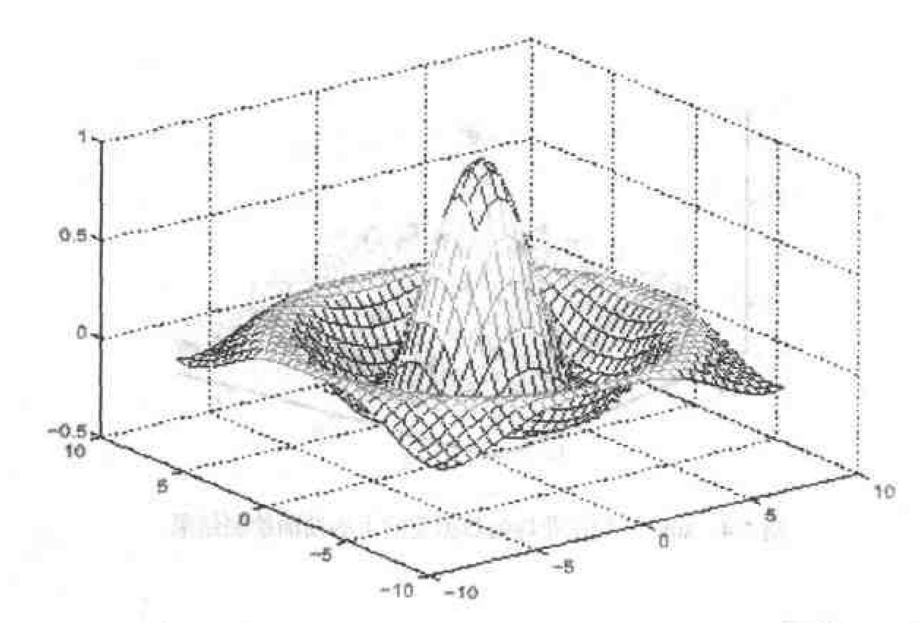


图 5-3　sinc 函数均匀数据空间上的曲面绘制结果

为了使用简单的非均匀数据生成曲面网格，首先使用 griddata 函数在非均匀空间点处进行插值，然后按通常的绘图方法调用 mesh 和 sruf。下例给出了在指定范围内的随机点上计算和绘制 sinc 函数曲线的方法。

用户数据为：

```
x = rand(100,1)*16-8;
y = rand(100,1)*16-8;
r = sqrt(x.^2 + y.^2) + eps;
z = sin(r)./r;
```

整个绘图过程包括以下几个步骤：

（1）使用 linspace 在用户的非均匀简单数据范围内生成均匀空间数据：

```
xlin = linspace(min(x),max(x),33);
ylin = linspace(min(y),max(y),33);
```

（2）根据 linspace 的输出调用 meshgrid 生成绘图网格：

```
[X,Y] = meshgrid(xlin,ylin);
[X,Y] = meshgrid(xlin,ylin);
```

（3）使用 griddata 根据原始非均匀数据点处的函数取值求 linspace 生成的均匀空间数据点处的函数插值，这一步是绘图的关键步骤。本例使用基于三角的 cubic 插值方法：

```
Z = griddata(x,y,z,X,Y,'cubic');
```

（4）使用绘图函数显示数据：

```
mesh(X,Y,Z)
axis tight;
hold on
plot3(x,y,z,'.','MarkerSize',15)
```

绘图结果如图 5-4 所示，图中红色的点表示给出的随机数据。

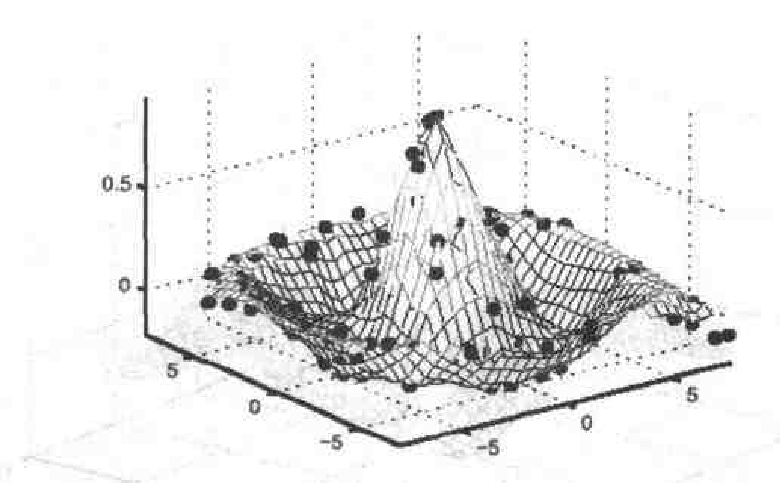


图 5-4　sinc 函数在非均匀数据空间上的曲面绘制结果

5.2.3　曲面特征设置

曲面对象属性提供了控制曲面外观的方法，用户可以通过设置相应的属性来定义边缘线形、节点标示、颜色、光源等等。除此之外，MATLAB 还提供一些能够改变曲面图形外观的技术，例如设置镜头位置和方向、添加光源、改变视点和视角、增加物体透明度以及缩放视图等。这些技术的使用方法将在下一章中详细说明。

为了说明这些技术对曲面图形效果的影响，使用以下语句给图 5-4 所示的曲面添加光源并调整视点（利用函数 daspect，axis，camlight，view），最后可以得到如图 5-5 所示的效果。显然，虽然图 5-5 中的 sinc 函数与图 5-3 中使用相同的数据，但后者的可视程度要好得多。

```
surf(X,Y,Z,'FaceColor','interp', 'EdgeColor','none',...
    'FaceLighting','phong')
daspect([5 5 1])
axis tight
view(-50,30)
camlight left
```

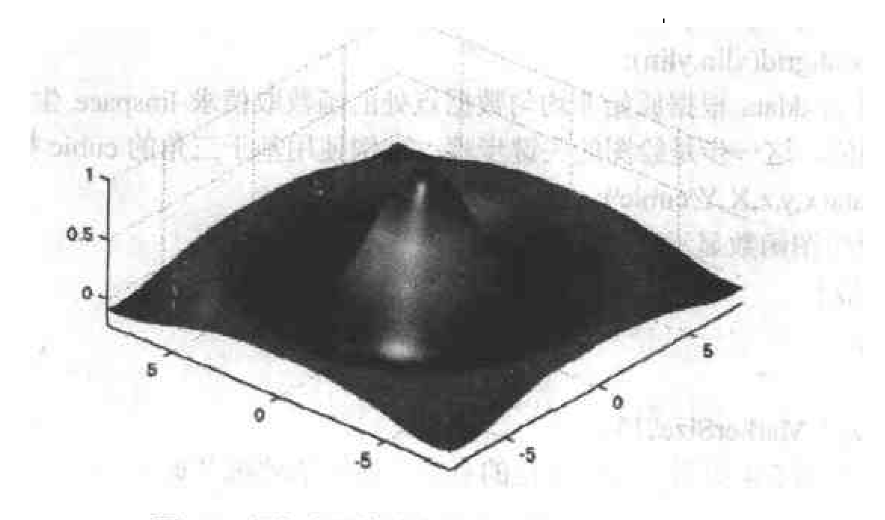


图 5-5　添加光源并改变视角后的曲面效果

5.2.4　曲面着色方法

用户可以通过控制 MATLAB 的着色方式来增强曲面图形的信息量。绘制曲面的函数不但能够指定绘图数据 x，y，还可以通过使用两个额外的向量或矩阵参数来描述曲面。如果 Z 是一个 m×n 矩阵，x 是 n 维向量，y 是 m 维向量，则：

mesh(x,y,Z,C)

将生成这样一个网格曲面：各网格的节点为(x(j),y(i),z(i,j))，颜色为 C(i,j)。可见，在定义了 C 矩阵之后，MATLAB 将不再采用默认的调色板，而是根据给定的颜色矩阵来绘制曲面网线。

更为通常的情况下，如果 X，Y，Z 和 C 都是同维的矩阵，那么：

mesh(X,Y,Z,C)

将生成这样一个网格曲面：其网格节点坐标为：(X(i,j),Y(i,j),Z(i,j)),颜色为 C(i,j)。

针对矩阵 C 类型的不同，MATLAB 将采取以下两种不同的曲面着色技术之一：

- 颜色索引：MATLAB 通过给每一个点指定一个调色板的索引来为曲面着色；
- 真彩：MATLAB 使用明确指定的颜色，即 RGB 值，来为曲面着色。

MATLAB 具体使用何种着色方式依赖于用户指定的颜色数据矩阵 C 的类型（单值或 RGB 值）。如果用户不明确定义颜色数据矩阵 C，那么 MATLAB 将根据数据的 z 坐标值来生成数据点相应的调色板下标；如果用户指定一个与 z 轴数据维数相等的矩阵 C 作为颜色数据，那么 MATLAB 将该数组作为调色板的下标；如果用户指定一个 m×n×3 的矩阵 C 作为颜色数组，则 MATLAB 视该图形为真彩图形，采用矩阵 C 的数据值作为点的 RGB 值。

这里给出一个曲面着色实例，该实例中颜色数据矩阵 C 的维数与 z 轴维数相等。使用球坐标来绘制一个球体，并根据 MATLAB 的一个特殊函数 hadamard 的符号来给球体着色。向量 theta 和 phi 取值范围分别是[-π，π]和[-π/2, π/2]。由于 theta 是行向量而 phi 是列向量，所以乘法运算的结果是三个矩阵 X，Y，Z。

```
k = 5;
n = 2^k-1;
theta = pi*(-n:2:n)/n;
phi = (pi/2)*(-n:2:n)'/n;
X = cos(phi)*cos(theta);
Y = cos(phi)*sin(theta);
Z = sin(phi)*ones(size(theta));
colormap([0 0 0;1 1 1])
C = hadamard(2^k);
surf(X,Y,Z,C)
axis square
```

绘制结果如图 5-6 所示。

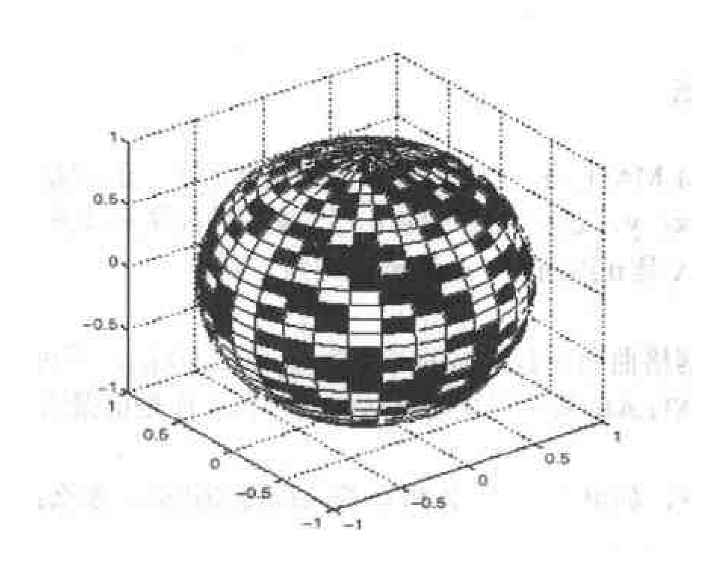


图 5-6　指定颜色索引矩阵的曲面着色效果

另外，缺省情况下，即使图形表面没有着色，MATLAB 也要隐藏视图中的部分不可见的网线。例如图 5-4 中所示的曲面图形，虽然并没有对该曲面的小面着色，但是 MATLAB 还是隐藏了曲面背面不可见的部分网格线。用户可以通过

hidden off

语句来迫使网格图形表面透明，从而显示所有位置处的网格线。图 5-7 显示了所有网格线均可见时的曲面显示效果。

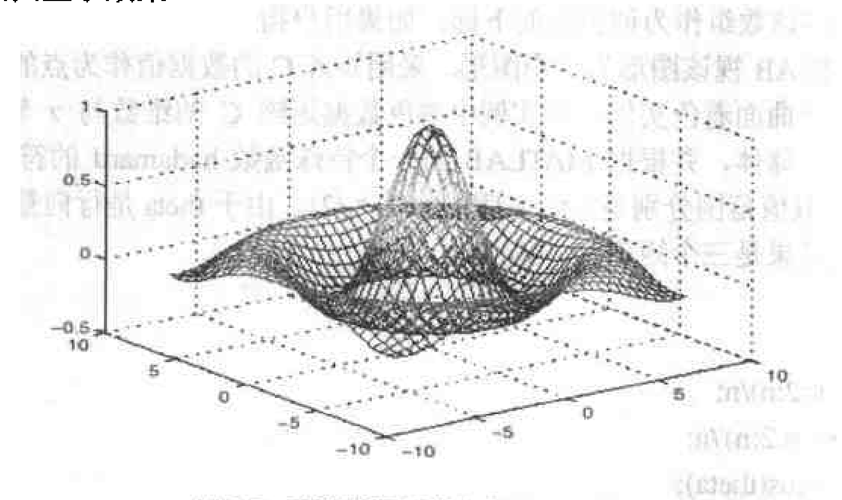


图 5-7　不隐藏网格线的曲面显示效果

5.2.5　调色板

每一个 MATLAB 图形窗口都具有一个调色板。调色板是一个长度等于定义颜色数目的列数为 3 的矩阵，该矩阵的每一行通过指定三个[0，1]范围内的数值元素来定义一个特殊颜色。实际上这些数值就是定义颜色的 RGB 成分。

无输入参数的函数 colormap 将返回当前的调色板。例如，MATLAB 缺省的调色板是 64 色的，第 57 种颜色是红色。表 5-1 列出了调色板中一些常用颜色的对应数值。

表 5-1 常用颜色与其对应数值

颜色	R 分量(红色)	G 分量（绿色）	B 分量（蓝色）
黑色	0	0	0
白色	1	1	1
红色	1	0	0
绿色	0	1	0
蓝色	0	0	1
黄色	1	1	0
洋红色	1	0	1
蓝绿色	0	1	1
灰色	0.5	0.5	0.5
暗红	0.5	0	0
铜色	1	0.62	0.40
碧绿色	0.49	1	0.83

用户可以使用 MATLAB 数组操作方法来创建调色板，也可以使用任意一个能够生成有效映射的函数（包括 hsv，hot，cool，summer 和 gray）来生成调色板。每一个函数都可以指定调色板的大小，例如：

hot(m)

将创建一个 m×3 的矩阵，该矩阵行元素指定 RGB 的亮度将按照黑、红、橙、黄、白的顺序逐渐变化。

使用 colorbar 函数可以在图形窗口中沿着用户图形方向显示当前的调色板。例如，以下语句：

```
[x,y] = meshgrid([-2:.2:2]);
Z = x.*exp(-x.^2-y.^2);
surf(x,y,Z,gradient(Z))
colorbar
```

将生成图 5-8 所示的带有竖直色带的曲面图形。

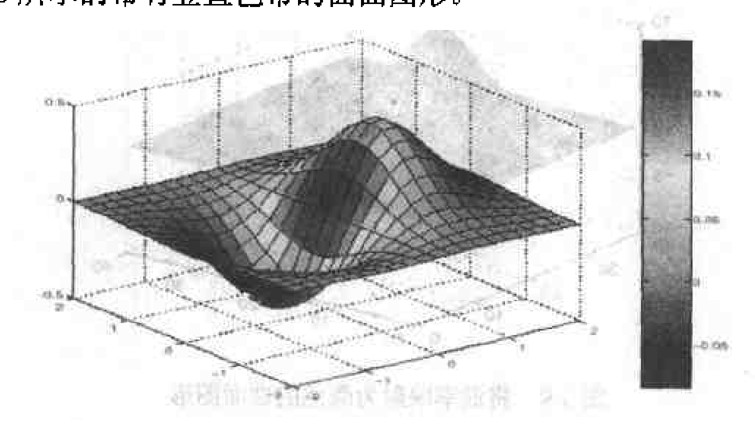


图 5-8 带有调色板色带的曲面图形

MATLAB 使用两种不同的映射方法将非真彩的颜色数据映射到调色板上：直接或按比例映射。

直接映射直接使用颜色数据作为调色板的下标，例如，数值为 1 的数据将映射到调色板

的第一个颜色，数值为 2 的映射到第二个颜色，以此类推。如果颜色数据不是整数，MATLAB 将取比该数值小的最接近的整数。数值大于映射表长度的颜色数据被映射到调色板的最后一个颜色，数值小于 1 的被映射到映射表的第一项。

按比例映射根据一个两元素的向量[cmin,cmax]（由 caxis 命令指定）来控制颜色数据到调色板的映射关系。cmin 指定映射到调色板第一个颜色的颜色数据值，cmax 定义映射到映射表最后一个颜色的颜色数据值。在这两个元素之间的数据将被线性地映射到映射表第二至倒数第二个颜色上，映射表达式为：

```
colormap_index = fix((color_dat-cmin)/(cma-cmin)*cm_length)+1
```

其中 cm_length 是调色板的长度。

缺省情况下，MATLAB 将根据坐标轴中图形对象的颜色数据跨度来定义[cmin,cmax]。用户也可以将该向量设置为任意数值，这就使得用户可以在同一个图形窗口中通过给每一个坐标轴定义一个不同比例的图形窗口映射表来同时显示多个坐标轴。

在缺省情况下，MATLAB 选择按比例映射的方法。如果希望使用直接映射，用户可以在创建图形时关闭按比例映射功能。例如：

```
surf(Z,C,'CDataMapping','direct')
```

下面给出一个将曲面的曲率映射为颜色的例子。一个曲面的拉氏算子与该曲面的曲率有关，可以使用函数 del2 来计算任意矩阵的离散拉氏算子。例如，使用 del2 来确定函数 peaks 返回数据的颜色：

```
P = peaks(40);
C = del2(P);
surf(P,C)
colormap hot
```

以上语句将通过计算数据的拉氏算子来创建一个颜色数组，这就使得曲面中曲率相同的地方使用相同的颜色。绘图结果如图 5-9 所示。

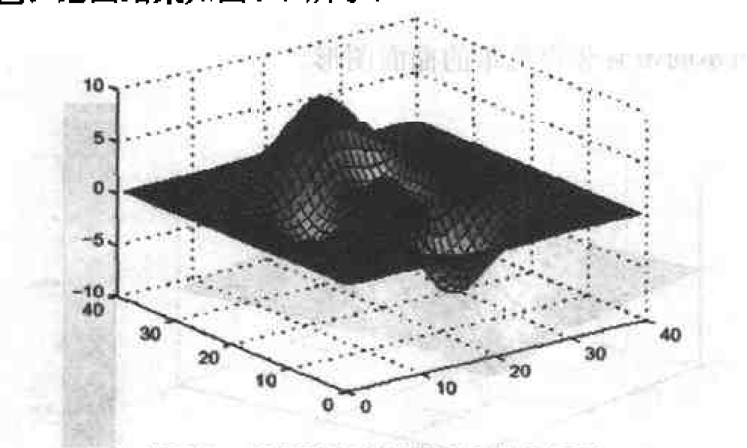


图 5-9　将曲率映射为颜色的曲面图形

如果不明确指定颜色数据，MATLAB 将使用 z 坐标值作为颜色数据进行颜色映射。例如，以下语句将生成一个与图 5-10 形状相同，但 z 坐标相同的地方颜色相同的曲面图形（如图 5-10 所示）：

```
surf(P);
colormap hot
```

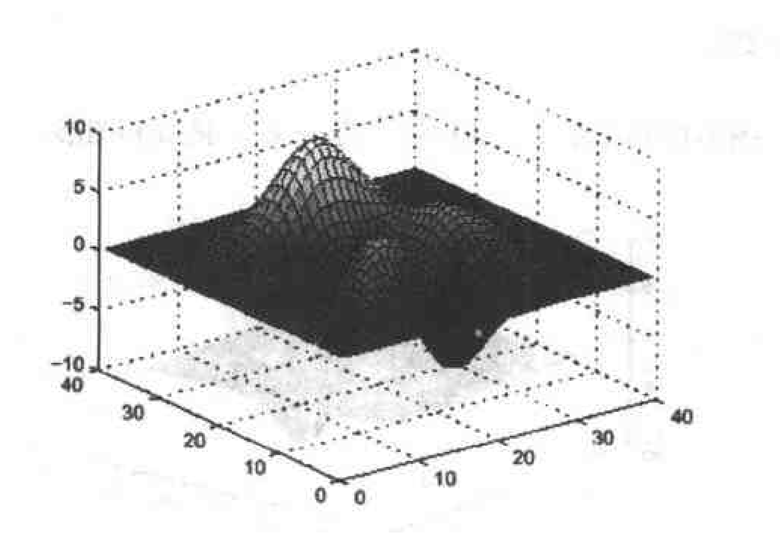


图 5-10 不指定颜色数据的曲面图形

由于调色板也是一个矩阵，因而用户可以对其进行任何矩阵操作，所以可以利用函数 bright 通过操作调色板矩阵来增强或减弱颜色的亮度。使用 rgbplot 函数显示调色板中颜色的 R，G，B 分量就可以看出 bright 函数作用。图 5-11 给出了使用 bright 函数产生的影响（以铜色为例）：

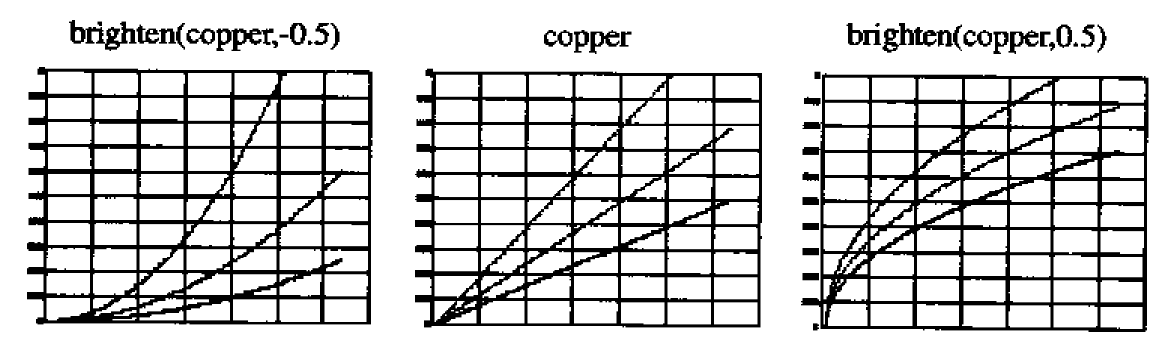


图 5-11 bright 函数对铜色产生的亮度影响

另外，根据电视信号的亮度成分 NTSC 颜色编码方法

亮度 = .30*R + .59*G + .11*B

使用非线性灰度映射

colormap([b b b])

可以有效地将一幅彩色图形转换为 NTSC 黑白等效图。

5.2.6 真彩图形

使用 24 位显示的计算机系统能够显示 2^{24} 个颜色，用户可以利用这个优势直接使用 RGB 值来定义颜色，省略调色板映射这一步骤。

使用一个 m×n×3 的矩阵来定义真彩，例如，以下语句：

```
Z = peaks(25);
C(:,:,1) = rand(25);
C(:,:,2) = rand(25);
```

```
C(:,:,3) = rand(25);
surf(Z,C)
```

将创建一个如图 5-12 所示的使用随机颜色的 peaks 矩阵真彩图形。

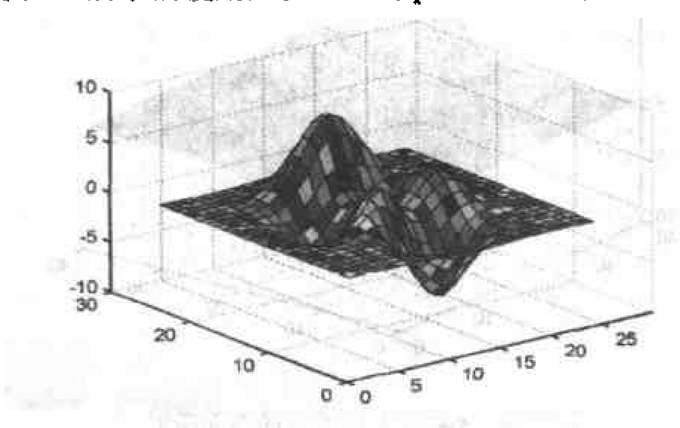


图 5-12　使用随机颜色的真彩图形

缺省情况下，MATLAB 使用 zbuffer 着色方法来显示真彩图形。如果用户将图形的着色器属性（Renderer）设置为 painter 并试图使用真彩定义一个图像、面片或曲面对象，MATLAB 将发出警告，同时不对物体进行任何着色。

用户也可以在非 24 位显示模式的计算机上显示真彩图，在这种情况下，MATLAB 使用一个特殊的调色板——抖动映射表——来产生尽可能逼近的颜色。

5.2.7　纹理映射

纹理映射通过对颜色数据的变换使之符合曲面绘图要求来实现二维图像到三维图形的映射。允许用户对一个曲面直接使用纹理（例如木纹）而无须计算使用纹理绘制曲面所需的几何模型。

纹理映射允许颜色数据的维数与曲面绘图数据的维数不符。用户可以使用任意尺寸的图像作为任何曲面的纹理。MATLAB 自动修改纹理颜色数据并将修改后的数据映射到整个曲面上。以下语句首先使用 sphere 函数生成一个球面，然后使用地球图像作为曲面的纹理。由于该地球图像仅是地球的一面，因此仅将图像映射到球面的一半上。

为了使用纹理映射，首先将 FaceColor 属性设置为 texturemap，然后将图像分配到曲面的 CData 中去：

```
load earth
sphere;
h = findobj('Type','surface');
hemisphere = [ones(257,125), X, ones(257,125)];
set(h,'CData',flipud(hemisphere),'FaceColor','texturemap')
colormap(map)
axis equal
view([90 0])
set(gca,'CameraViewAngleMode','manual')
```

```
view([65 30])
```

以上语句产生的纹理映射结果如图 5-13 所示。

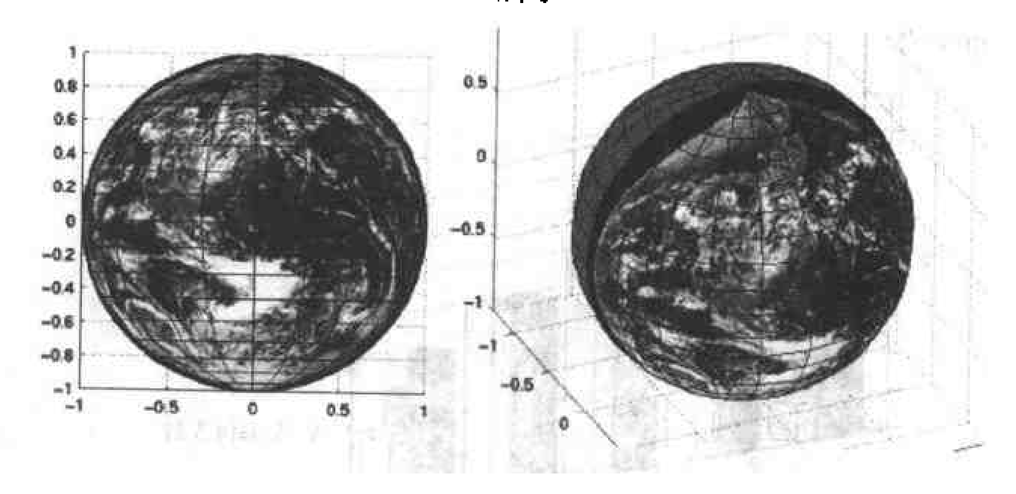


图 5-13　将左图作为纹理映射到球体的显示效果

5.3　特殊三维图形

5.3.1　三维直方图

使用函数 bar3 来绘制三维直方图。三维直方图可以分为两种：分离直方图和聚合直方图。分离直方图使用函数 bar3 的最简形式 bar3(Y)来获得，该语句能够将矩阵 Y 每一个元素绘制成分离的三维直方块，每一列的元素沿 y 轴分布。表示矩阵第一列元素的直方块沿 y 轴分布，都位于 x=1 处；表述最后一列元素的直方块沿 y 轴分布，都位于在 x=3 处。例如：

```
bar3(Y)
```

将显示如图 5-14 所示的沿 y 轴的五群（每群三个）直方块组。

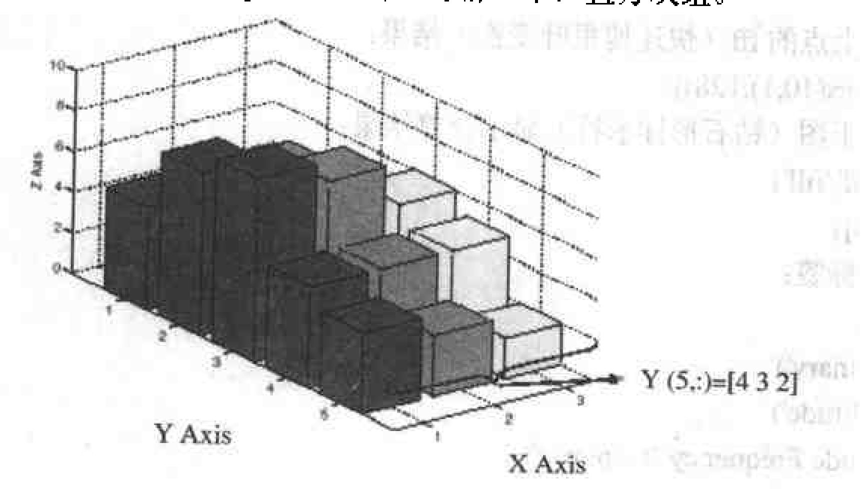


图 5-14　分离三维直方图

使用函数 bar3 的参数 group 可以将三维直方块按行向量分丛，每一丛沿 y 轴均匀分布，从而获得聚合的三维直方图。例如，以下语句将产生如图 5-15 所示的结果。

```
bar3(Y,'group')
```

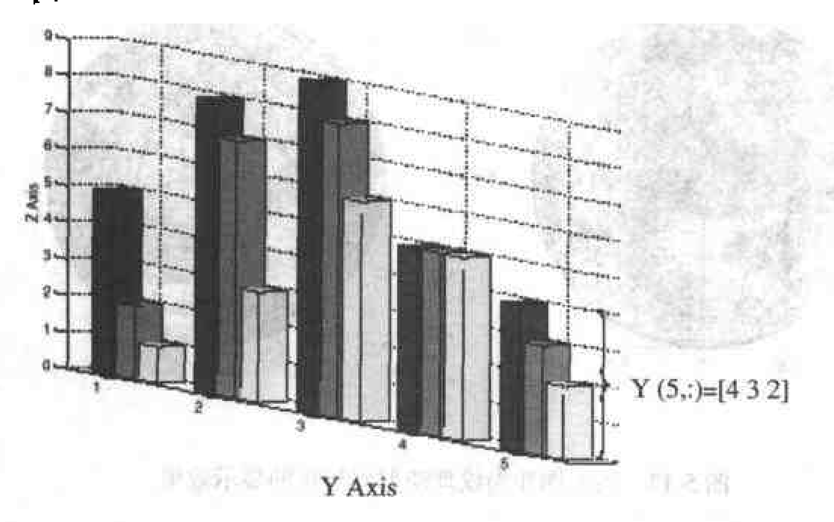


图 5-15　聚合三维直方图

5.3.2　三维枝干图

函数 stem3 绘制在 x-y 平面上扩展的三维枝干图。如果该函数只有一个向量输入参数，MATLAB 将首先判断该向量是行向量还是列向量，然后将枝干图绘制在 x=1 或 y=1 处。下面将举例说明该函数的使用方法。

假设需要对复平面单位圆上的点进行快速傅立叶变换。为了清楚地描述变换后的效果，将单位圆上的点进行可视化处理。首先计算单位圆：

```
th = (0:127)/128*2*pi;
x = cos(th);
y = sin(th);
```

计算单位圆上点的 fft（快速傅里叶变换）结果：

```
f = abs(fft(ones(10,1),128));
```

使用三维枝干图（钻石形标示符）显示计算结果：

```
stem3(x,y,f','d','fill')
view([-65 30])
```

给图形添加标签：

```
xlabel('Real')
ylabel('Imaginary')
zlabel('Amplitude')
title('Magnitude Frequency Response')
```

改变视图方向：

```
rotate3d on
```

绘图结果如图 5-16 所示。

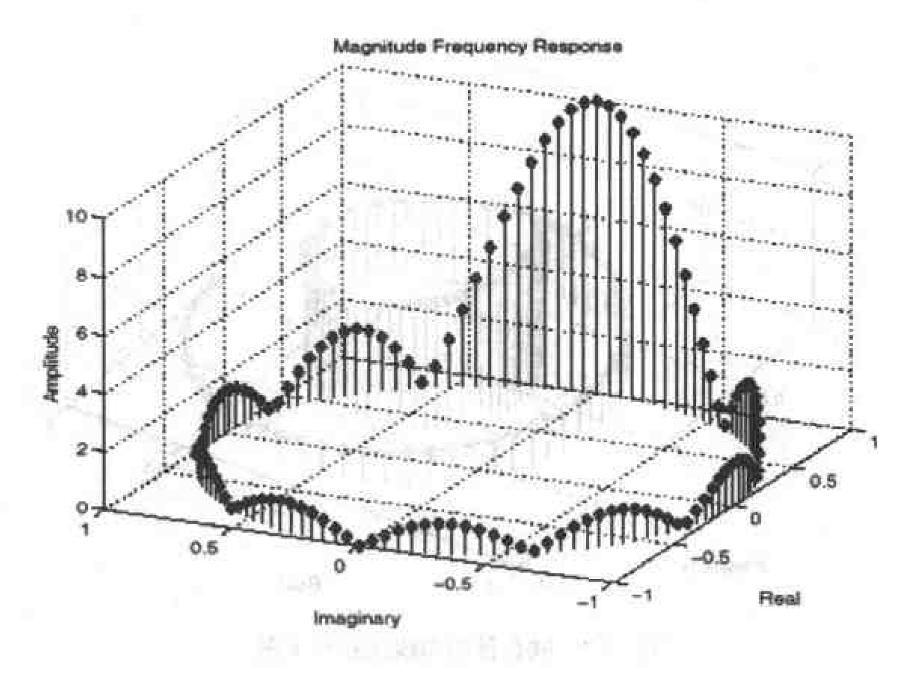


图 5-16 三维枝干图

联合使用曲线图和枝干图是一种非常有效的数据描述方法。下面给出一个实例来说明这一点。

使用 stem3 来可视化基本函数

$$y = e^{-st}$$

再指定常数 s 处的拉氏变换：

```
t = 0:.1:10;      %时间范围
s = 0.1+i;        %螺旋率
y = exp(-s*t);    %计算衰减指数
```

以 t 为时间增量创建一系列高度渐增的枝干曲线，然后使用一条曲线将枝干图的顶部连接起来以便于观察：

```
 stem3(real(y),imag(y),t)
 hold on
 plot3(real(y),imag(y),t,'r')
 hold off
view(-39.5,62)
```

添加标签：

```
xlabel('Real')
ylabel('Imaginary')
zlabel('Magnitude')
```

绘图结果如图 5-17 所示。

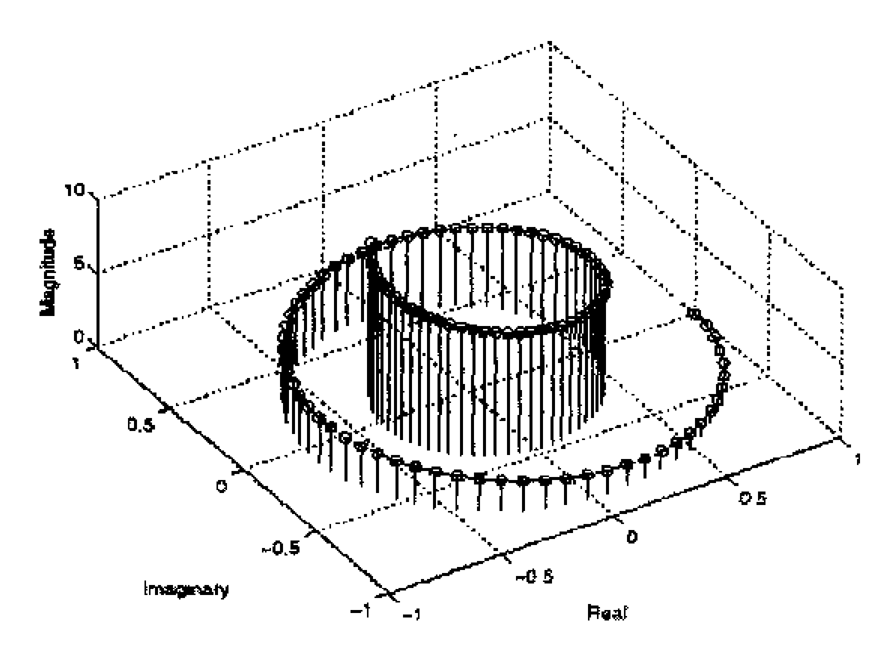


图 5-17　联合使用曲线图和枝干图

5.3.3　三维箭头图形

三维箭头图形可以在(x,y,z)位置处显示由(u,v,w)分量组成的向量。例如，可以用一个时间函数来表明导弹发射路径：

$$z(t)=v_Z t+\frac{at^2}{2}$$

首先定义常数：

```
vz = 10; % 速度
a = -32; % 加速度
```

然后计算 t 在 0 到 1 之间以 0.1 为间隔变化时的函数值（高度）z：

```
t = 0:.1:1;
z = vz*t + 1/2*a*t.^2;
```

计算 x 和 y 方向的位置：

```
vx = 2;
x = vx*t;
vy = 3;
y = vy*t;
```

计算速度向量的各个分量并使用三维箭头图形显示该向量：

```
u = gradient(x);
v = gradient(y);
w = gradient(z);
scale = 0;
quiver3(x,y,z,u,v,w,scale)
```

axis square

绘制结果如图 5-18 所示。

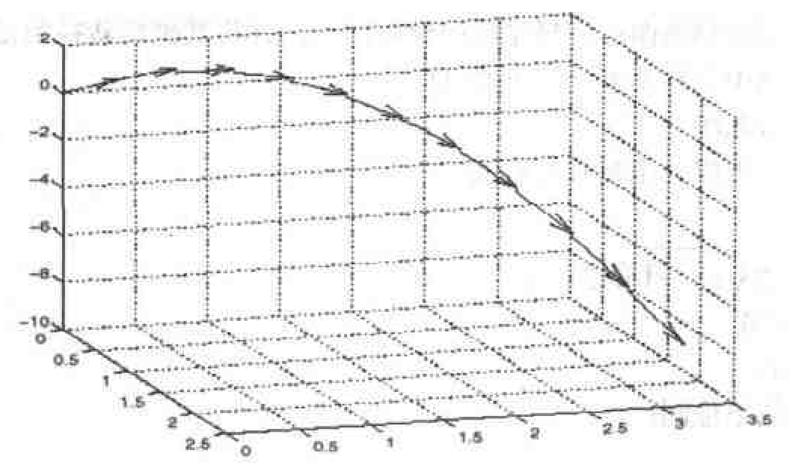


图 5-18　三维箭头图形

5.3.4　三维等值线图形

使用 contour3 函数显示由给定矩阵 Z 生成的三维等值线。下面举例说明该函数的使用方法。

使用以下语句将在笛卡尔坐标系中绘制 peaks 函数产生的等值线：

```
[X,Y,Z] = peaks;
contour3(X,Y,Z,20)
h = findobj('Type','patch');
set(h,'LineWidth',2)
title('Twenty Contours of the peaks Function')
```

显示结果为一幅包含 20 条等值线、线宽为两个点的三维图形（如图 5-19 所示）。

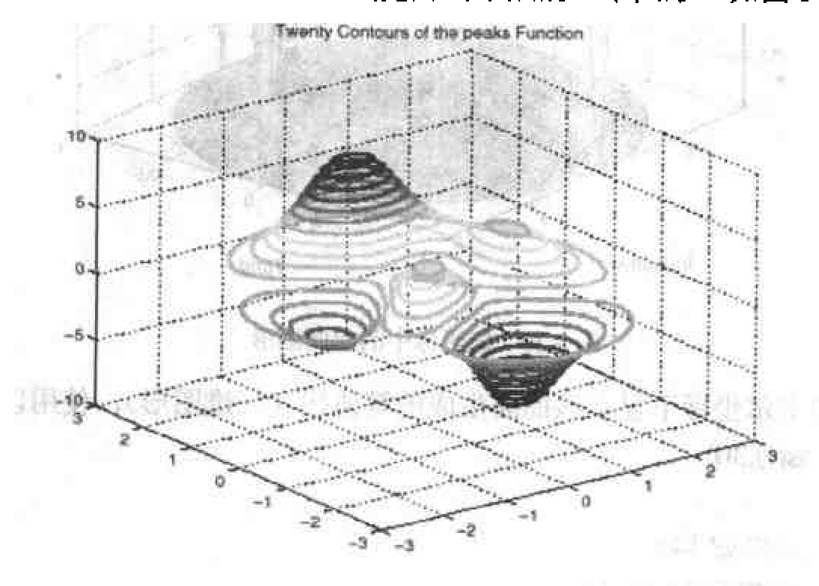


图 5-19　笛卡尔坐标系中的函数等值线图形

除了绘制给定函数的等值线，函数 contour3 还可以计算并绘制由给定曲面生成的等值线。下面将分三个步骤来说明如何在极坐标中显示一个由曲面生成的等值图形。

第一步：首先要绘制曲面：在极坐标中建立网格，并将其坐标转换为笛卡尔坐标：

```
[th,r] = meshgrid((0:5:360)*pi/180,0:.05:1);
[X,Y] = pol2cart(th,r);
```

然后在单位圆内生成一个复数矩阵Z：

```
Z = X+i*Y;
```

创建并显示函数 $\sqrt[4]{z^4-1}$ 的曲面：

```
f = (Z.^4-1).^(1/4);
surf(X,Y,abs(f))
```

显示该曲面下的单位圆：

```
hold on
surf(X,Y,zeros(size(X)))
hold off
```

添加标签：

```
xlabel('Real','FontSize',14);
ylabel('Imaginary','FontSize',14);
zlabel('abs(f)','FontSize',14);
```

绘制结果如图5-20所示。

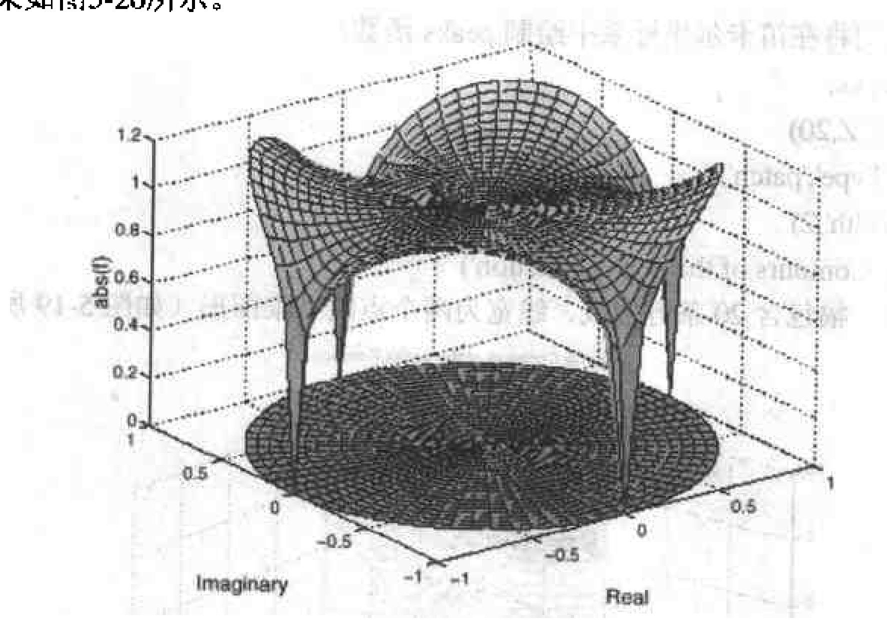


图 5-20　极坐标中的曲面图形

第二步：在笛卡尔坐标下显示该曲面生成的等值线（二维图形），使用以下语句：

```
contour(X,Y,abs(f),30)
axis equal
xlabel('Real','FontSize',14);
ylabel('Imaginary','FontSize',14);
```

绘图结果如图 5-21 所示。

第三步：使用极坐标轴来显示曲面的等值线。首先使用 polar 函数创建一个极坐标轴，然后删除 polar 指定的线条：

```
h = polar([0 2*pi], [0 1]);
delete(h)
```

显示等值线网格：

```
hold on
contour(X,Y,abs(f),30)
```

结果如图 5-22 所示。

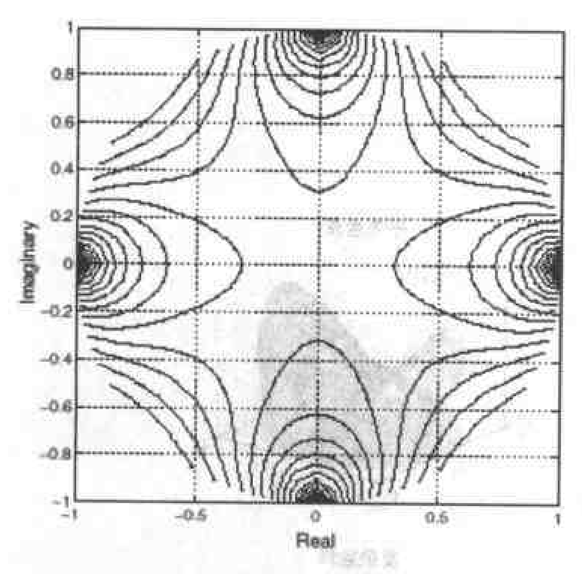


图 5-21　笛卡尔坐标系下的曲面二维等值线图

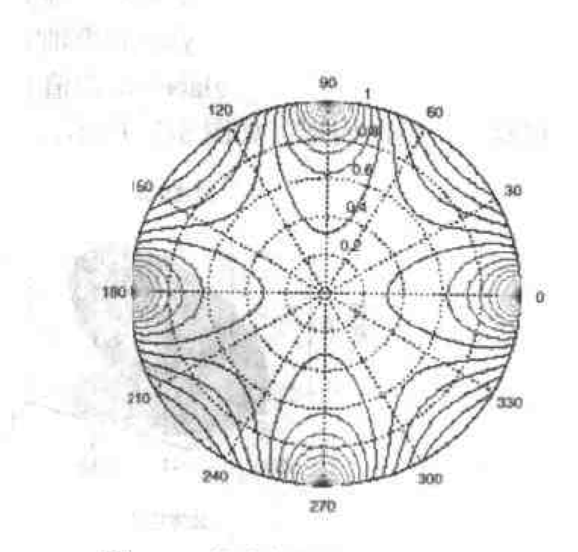


图 5-22　极坐标下的曲面二维等值线

5.4　实 例 讲 解

【实例】：如何在极坐标表下绘制复数函数 z、z^2、z^3、**和** $\sqrt[4]{z^4-1}$ **的曲面图形？**

分析：以上的内容重点介绍了实数向量和矩阵及函数的图形绘制，本问题则要求绘制复数的函数图形。为了实现这一要求，使用专门的复数绘图网格和绘图函数，这些函数的用法和以上介绍的绘图函数没有本质的区别，只是输入参数类型有所不同。

解决方法：

步骤一：准备网格。使用 cplxgrid 函数创建一个 21×41 的复数网格矩阵：

```
z=cplxgrid(20);
```

步骤二：将窗口分割成四个子图形，在每一个子图形中使用 clpxmap 函数绘制相应的复数函数的极坐标曲面图形，标注左上和右下图形的坐标轴并添加每个子图形的标题：

```
subplot(2,2,1),cplxmap(z,z)
        title('线性运算')
        xlabel('实轴')
```

```
                    ylabel('虚轴')
                    zlabel('函数值')
subplot(2,2,2),cplxmap(z,z.^2)
                     title('平方运算')
subplot(2,2,3),cplxmap(z,z.^3)
                     title('立方运算')
subplot(2,2,4),cplxmap(z,(z.^4-1).^1/4)
                    title('复杂运算')
                    xlabel('实轴')
                    ylabel('虚轴')
                  zlabel('函数值')
```

步骤三：输出图形：如图 5-23 所示。

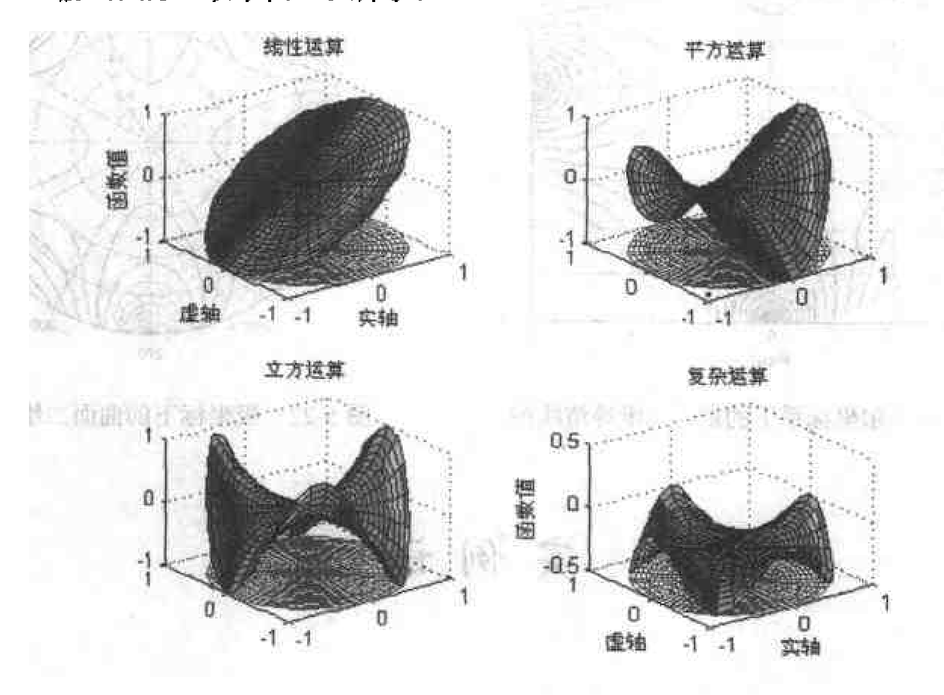


图 5-23　极坐标下的复数函数曲线图形

5.5 小　　结

通过本章的学习，读者可以学会灵活使用三维绘图函数以及图形属性进行数据绘制，使数据具有一定的可读性，能够表达一定的信息。在以后的学习过程中，读者将学习如何使用 MATLAB 的各种三维可视化技术来进一步增强三维图形的真实感。

第 6 章 MATLAB 三维可视化技术

在掌握了三维图形绘制方法的基础上，本章将向读者介绍一些增加三维图形真实感的技术。首先向读者介绍有关术语和数据可视化的基本过程，然后介绍定义视点、使用镜头对象、使用灯光对象、利用对象的透明属性等可视化实用技术的使用方法。通过对三维图形使用这些可视化技术，能够进一步提高数据的可读性，得到真实感较强的三维场景。

6.1 创建三维模型

6.1.1 基本术语

MATLAB 图形系统中有许多专用术语，大多都是针对三维图形而言的。这里简单介绍以下几个常用的基本术语：

可视化（Visualization）：使用图形信息描述方法能够更加清楚地描述信息的某种特征。可视化技术利用不同的几何形式和着色方法对数据进行某种特定方式的映射，使数据能够体现尽量多的信息。几何形式可以是实物，例如飞机，也可以是代表数据值的图形元素，例如流线；

节点（vertex）：节点是三维空间中的最小元素，在本书中特指网格的交点；

坐标轴（axes）：在使用计算函数时，坐标轴是指数轴，而在进行图形操作时，坐标轴是指图形操作的参考轴，注意此时坐标轴也可以是一个点；

镜头（camera）：这里是指用户观察曲面图形的空间位置；

灯光（light）：这里是指使用一个有向光源对物体进行照明的技术，这项技术能够使曲面的形状产生某种特殊的细微变化，从而更易于观察。灯光能够为三维图形增加真实感；

方位角（azimuth）：是指镜头在 x-y 平面中的极角，即视点（镜头所在的点）与原点间的连线在 x-y 平面上的投影与 x 轴所成的夹角。一个正的方位角标志着标准视图将向逆时针方向旋转某个角度；

高度角（elevation）：视点与原点的连线在 x-y 平面上的投影与 x-y 平面所成的角。高度角主要用来表明方位角的位置是在 x-y 平面的上方还是下方。

颜色映射表（colormap）：一个有序的颜色列表，通常用来指定非真彩图形中数据到颜色的映射范围。MATLAB 系统默认的颜色映射表是一个 64 色的颜色表。

6.1.2 创建三维场景基本步骤

为了详细说明创建三维场景（包括数据图和三维物体模型）每一步骤的作用，以下将以

一个具体实例为主进行介绍。该实例说明了如何绘制 peaks 函数具有一定真实感的三维彩色曲面。

步骤一：准备数据：

```
Z = peaks(20);
```

步骤二：选择窗口并确定绘图区域：

```
figure(1)
subplot(2,1,2)
```

步骤三：调用三维图形函数：

```
h = surf(Z);
```

步骤四：设置色彩映射和阴影算法：

```
colormap hot
shading interp
set(h,'EdgeColor','k')
```

步骤五：添加光源：

```
light('Position',[-2,2,20])
lighting phong
material([0.4,0.6,0.5,30])
set(h,'FaceColor',[0.7 0.7 0],'BackFaceLighting','lit')
```

步骤六：设置视点：

```
view([30,25])
set(gca,'CameraViewAngleMode','Manual')
```

步骤七：设置坐标范围和标尺：

```
axis([5 15 5 15 -8 8])
set(gca'ZTickLabel','Negative||Positive')
```

步骤八：设置外观比例：

```
set(gca,'PlotBoxAspectRatio',[2.5 2.5 1])
```

步骤九：用坐标标注图形：

```
xlabel('X Axis')
ylabel('Y Axis')
zlabel('Function Value')
title('Peaks')
```

步骤十：打印图形。

```
set(gcf,'PaperPositionMode','auto')
-dps2
```

绘制结果如图 6-1 所示。

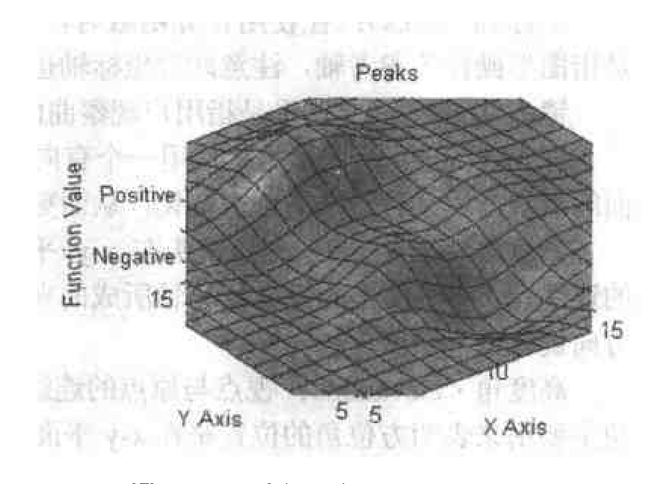


图 6-1　三维场景创建演示程序图形

6.1.3　使用面片创建三维模型

一个面片图形对象是由一个或多个多边形组成的，这些多边形之间可以是相连的，也可以是不相连的。面片对于真实物体建模（例如飞机、汽车等）以及绘制任意形状的二维或三

维多面体是非常有用的。一般的曲面对象都是由矩形网格组成的，适合于显示比较平坦的图形（例如数学函数图形），而面片则适于绘制矩形平面内的等高线数据以及物体的参数化表面（例如球体表面）。有这样几个函数可以用来创建面片对象：fill，fill3，isosurface，isocaps，contour以及patch。这里仅介绍patch函数的使用方法。

用户通过指定面片的多边形顶点坐标和颜色数据来定义一个面片。面片支持许多颜色选项，如果与多种不同的几何形状联合使用，就可以非常有效地完成数据的可视化工作。定义面片可以使用以下两种方法：一是用户定义每一个多边形的各个顶点坐标，由MATLAB将这些坐标连接起来形成面片；二是通过给出面片中所有各不相同的多边形顶点坐标以及这些顶点的连接关系矩阵来定义面片。第二种方法适合于定义多面面片，因为这种方法通常只需要很少的数据来定义面片，那些表面共享的顶点坐标仅需要定义一次。

patch函数有两种形式：高级语法形式和低级语法形式。当用户使用高级语法形式时，MATLAB将根据用户指定的颜色数据自动确定每一个表面的颜色。只要用户在指定x，y和z轴坐标以及颜色数据时按照以下的正确顺序，高级语法形式允许用户忽略各属性的名称，直接定义属性的取值：

```
patch(x-coordinates,y-coordinates,z-coordinates,colordata)
```

用户必须定义颜色数据使MATLAB知道将使用哪种颜色。因此，MATLAB总是将第三（当存在z坐标轴时为第四）个参数视为颜色数据。如果用户希望使用x，y，z坐标来定义面片而不定义颜色，MATLAB将会把z坐标数据理解为颜色数据，然后绘制二维面片。例如：

```
h = patch(sin(t),cos(t),1:length(t))
```

绘制一个所有顶点的z坐标为0的，每个顶点使用不同颜色的面片。而

```
h = patch(sin(t),cos(t),1:length(t),'y')
```

将绘制一个顶点z坐标值逐渐变大、颜色为黄色的面片。

patch函数的低级语法形式必须以属性名与属性值对作为输入参数，同时，除非用户改变了FaceColor属性值，否则MATLAB将调用缺省的FaceColor属性值（白色）对表面进行着色，此时用户是看不出着色效果的。例如，以下语句：

```
patch('XData',sin(t),'YData',cos(t))
```

将绘制一个白色表面的面片。

下面介绍面片的具体创建方法。首先介绍怎样创建一个简单的单面面片，即一个多边形。用以下语句来定义多边形的顶点坐标和颜色：

```
patch(x-coordinates,y-coordinates,[z-coordinates],colordata)
```

例如，以下语句将显示一个黄色表面、黑色边缘的十边形。axis equal使得该多边形按照标准的比例显示。

```
t = 0:pi/5:2*pi;
patch(sin(t),cos(t),'y')
axis equal
```

多边形的第一个和最后一个顶点不必重合，因为MATLAB将自动闭合面片的每一个面。

用户可以自己控制面片的着色方式（即颜色外观），例如，用户可以提供一组数值将每个顶点的颜色映射为图形窗口颜色映射表的某种颜色。

```
a = t(1:length(t)-1);
patch(sin(a),cos(a),1:length(a),'FaceColor','interp')
```

```
colormap cool;
axis equal
```

绘图结果如图6-2所示。

MATLAB现在将对面片每个面的颜色进行插值计算，同样可以使用插值方法对边缘进行着色。命令形式为：

```
patch(sin(t),cos(t),1:length(t),'EdgeColor','interp')
```

着色结果如图6-3所示。

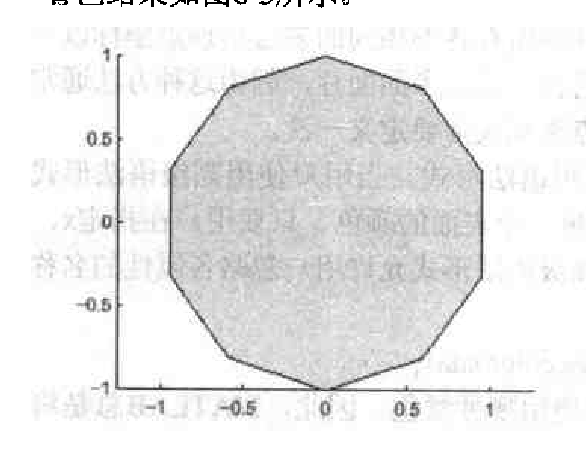


图 6-2　单一颜色的面片

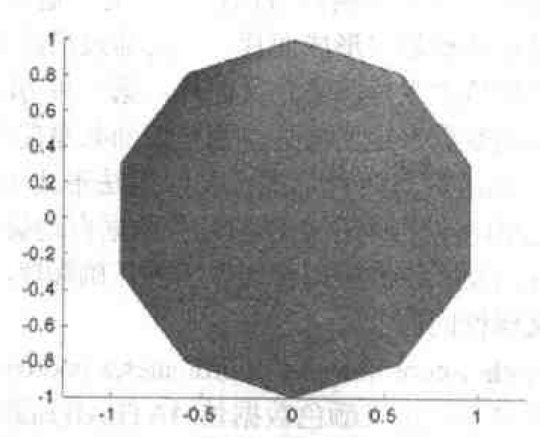


图 6-3　使用插值着色方式的面片

下面介绍如何创建有多个小面（多边形）的面片。如果用户指定x，y和z坐标轴参数为向量，那么MATLAB将通过连接各个顶点来绘制一个简单的多边形；如果参数是矩阵，那么MATLAB将根据矩阵的每一列绘制一个多边形，结果就会是一个简单的多小面的面片，注意这些小面可以是不相连的。

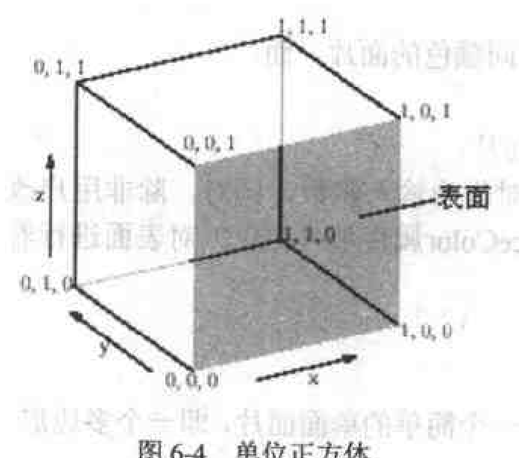


图 6-4　单位正方体

假设要定义一个如图6-4所示的单位正方体。

正方体每个面有六个顶点，由于无需封闭每个多边形（即第一个和最后一个顶点坐标可以不相同），所以可以使用三个4×6的矩阵来表示每个面每个顶点的x，y和z坐标值。由三个矩阵的相同列共同来表示正方体相应的面。在这种定义方式下，虽然正方体仅有8个顶点，但是用户必须定义24个点才能够定义正方体的六个面。这三个矩阵分别为：

x = 0 1 1 0 0 0；1 1 0 0 1 1；1 1 0 0 1 1；0 1 1 0 0 0

y = 0 0 1 1 0 0；0 1 1 0 0 0；0 1 1 0 1 1；0 0 1 1 1 1

z = 0 0 0 0 0 1；0 0 0 0 0 1；1 1 1 1 0 1；　1 1 1 1 0 1

前面已经提到，用户可以通过定义面片中所有独一无二顶点的坐标以及它们的连接顺序来构成面片。在这个正方体中，既然每个面都与其他四个面之间共享顶点，那么就可以使用这种方法来更为有效地定义面片。通过以下两个矩阵就可以定义该正方体：

顶点矩阵vertex_matrix=

0 0 0；1 0 0；1 1 0；0 1 0；0 0 1；1 0 1；1 1 1；0 1 1

连接关系矩阵faces_matrix=

1 2 6 5; 2 3 7 6; 3 4 8 7; 4 1 5 8; 1 2 3 4; 5 6 7 8

连接关系矩阵每一行中的数值分别表示顶点矩阵的行标，例如正方体的第一个面是由顶点矩阵的第一、二、六、五个行向量[0 0 0]、[1 0 0]、[1 0 1]和[0 0 1]构成的。

此时需要使用面片的Vertices和Faces属性来生成该面片，为此必须采用patch函数的低级语法格式，因为只有这样才能明确地定义Vertices和Faces属性值。另外，在这种形式下，用户还必须自己指定表面和边缘的颜色。使用Vertices和Faces属性将会节省大量内存，当面片包含许多个小面时，这一点尤为重要。本例中采用八色的hsv色彩映射方法对正方体的每个面进行插值着色，用户需要给每个顶点定义一个颜色并将FaceColor属性设置为interp。

```
patch('Vertices',vertex_matrix,'Faces',faces_matrix,...
      'FaceVertexCData',hsv(8),'FaceColor','interp')
```

将该图形转换为标准的三维视图并转换为球坐标形式：

```
view(3);
axis square
```

以上语句将生成如图6-5所示的正方体。

图6-5　定义颜色后的单位正方体

面片对象的颜色映射表与曲面对象的颜色表有本质区别：面片不像曲面对象那样根据顶点的z坐标给每个顶点自动生成颜色数据，用户必须为面片指定明确的颜色，或使用MATLAB缺省的白色表面和黑色边缘。使用面片颜色能够增强物体的真实感，提供更多的信息量。例如通过给飞机的两翼使用不同的着色方法可以表明两翼承受的不同压力。



用户可以使用三种方式指定面片表面的颜色：

- 所有表面使用单一颜色；
- 使用平铺着色方式为每个面定义一个颜色；
- 使用插值着色为每个顶点定义一个颜色。

这三种方式的具体使用方法与面片的具体属性有关。表 6-1 给出了控制面片颜色的属性及其含义。

表 6-1　面片颜色控制属性及其含义

属 性 名 称	含　义
CData	联合 x、y 和 z 数据定义每个面或顶点的颜色
CDataMapping	指定颜色数据是通过映射还是直接作为窗口色彩映射表的下标
FaceVertexCData	联合每个面和顶点的数据定义每个面或顶点的颜色
EdgeColor	定义边缘采用由顶点颜色决定的可见单平铺着色还是插值着色
FaceColor	定义表面采用由顶点颜色决定的可见单平铺着色还是插值着色
MarkerEdgeColor	指定标示的颜色或填充形标示的边缘颜色
MarkerFaceColor	指定填充型标示的填充颜色

面片的定义方式不同，着色时使用的属性也不同。如果使用 x，y，z 坐标来定义面片，那么要采用 CData 属性定义面片颜色；如果使用顶点和连接关系定义面片，则使用

FaceVertexCData 属性定义颜色。

面片的每一个面都有一个边界，边界定义颜色的方式有以下几种：

- 使用前一个顶点的颜色给边界平铺着色，例如给一个四面面片的边缘着色（表面不着色），效果如图 6-6 所示；
- 使用该边界的两个顶点的颜色给边界进行插值着色。在这种情况下，顶点的连接关系并不影响边界的颜色。

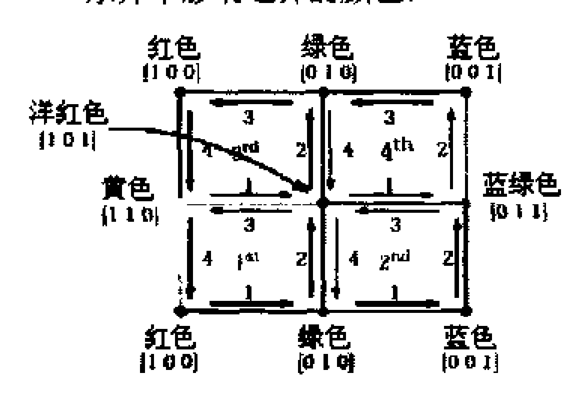


图 6-6　平铺着色方式

MATLAB 采用两种方法对颜色数据进行插值：一种是颜色数据索引：将数值映射到图形窗口颜色映射表中定义的某种颜色；另一种是数据真彩化：不使用颜色映射表，采用 RGB 值明确地定义颜色。颜色数据的位数决定了 MATLAB 的插值方式。如果仅给每个面、顶点或面片定义一个数值，那么 MATLAB 将采用颜色数据索引方式；如果给每个面、顶点或面片定义三个数值，则 MATLAB 将这三个值作为 RGB 值来使用。

使用颜色数据索引方式时，缺省情况下，MATLAB 将最小的颜色数据映射为颜色数据表的第一个颜色，最大的映射为最后一个颜色，其他数据则线性地扩展到整个映射表中。这样，用户就可以使用任意范围的数据来调用颜色映射表而无需对数据进行转换。例如，以下面片具有 8 个三角形表面，24 个顶点，3×8 矩阵变量 c 表示颜色数据，c 的每一列定义一个面三个顶点的索引颜色。

```
c =
  1 4 7 10 13 16 19 22
    2 5 8 11 14 17 20 23
    3 6 9 12 15 18 21 24
```

绘制这个面片将会得到如图6-7所示的结果。

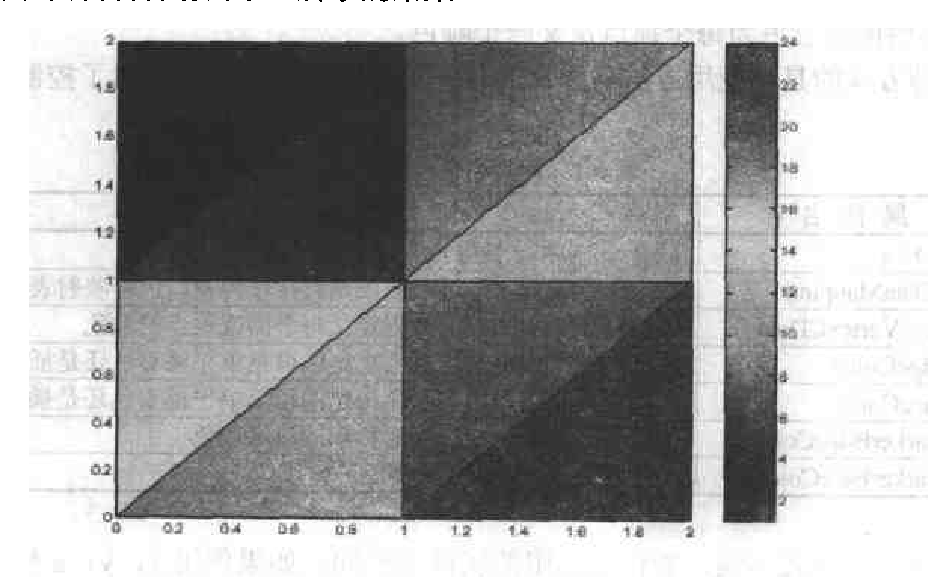


图 6-7　缺省颜色索引方式效果

可以使用caxis命令改变颜色数据的映射方式。该命令使用一个二元素的向量[cmin，cmax]指定映射到映射表首尾的数据值。缺省情况下，MATLAB将cmin设置为坐标轴中所有图形对象颜色数据的最小值，cmax为最大值。颜色数据可以不是一列连续的整数，它可以是与数据坐标轴维数相符的任意矩阵。例如以下语句：

```
patch(x,y,z,rand(size(z)))
```

如果用户将patch函数的CdataMapping属性设置为off：

```
set(patch_handle,'CDataMapping','off')
```

那么MATLAB将每一个颜色数值理解为颜色映射表的下标，也就是说，数值1对应映射表的第一个颜色，数值2对应第二个颜色，依此类推。假如上例使用这种理解方式，那么绘图结果将会发生一定变化（如图6-8所示）。

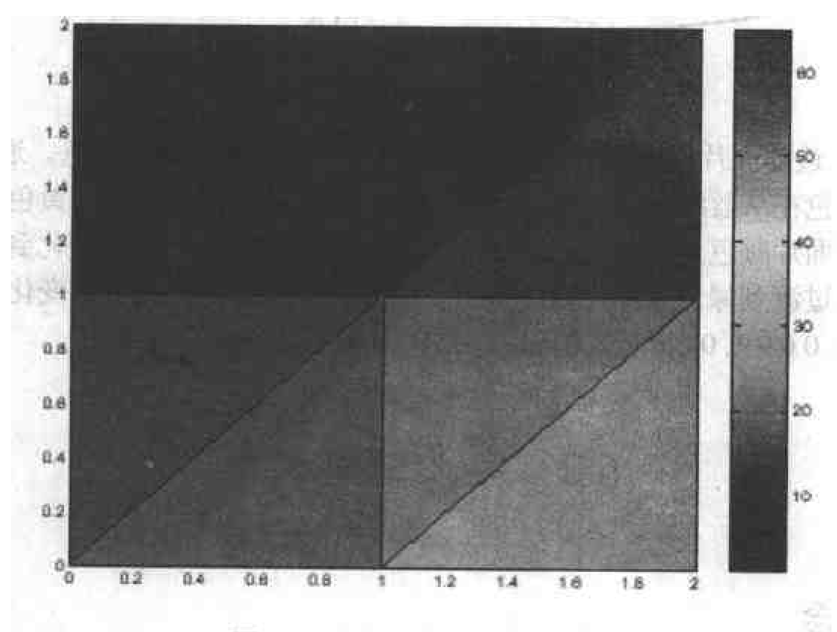


图6-8　改变颜色索引方式的效果

在这种方式下，如果给出的颜色数值不是整数，MATLAB将对数据作如下的变换：如果数值小于1，则将该数值映射为颜色映射表的第一个颜色；如果数值大于映射表的长度，则将该数值映射为映射表的最后一个颜色；对其他非整数做取整运算，即舍弃小数点后的位数。

真彩是指使用明确的RGB值取代图形窗口颜色映射表来指定一个颜色。真彩能够提供比颜色映射表更大范围的颜色。使用真彩将消除数据到映射表的映射入口，因而如果不重新定义颜色数据的数值，那么面片的颜色也就不能改变。

当用户修改面片的颜色数据时，MATLAB将通过修改每个顶点的颜色来确定每个面的颜色。修改方式依赖于用户是否定义了真彩数据或索引颜色数据。如果使用的是真彩数据，MATLAB将修改每个顶点的RGB值，这通常会在面片的小面上产生均匀变化的颜色。比较而言，索引颜色数据对应的面片仅仅使用颜色映射表中的颜色，显示的效果会因为映射表的不同而大大不同。为了说明这一点，下面给出两个使用相同顶点数据定义的面片，圆圈标记表示黄色、红色和蓝色顶点颜色，如图6-9所示。

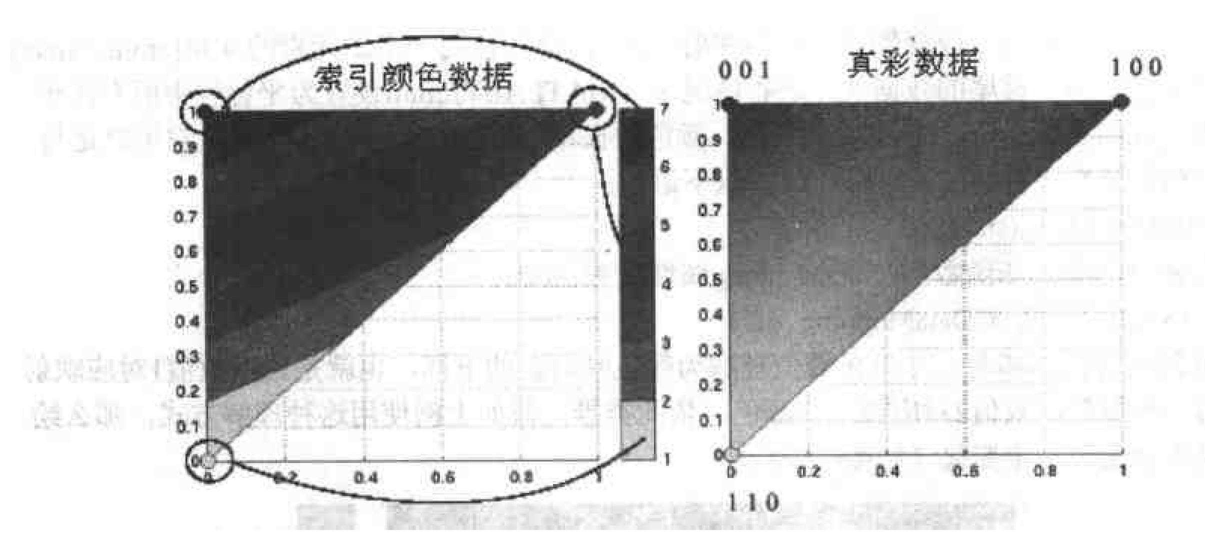


图 6-9 真彩与索引颜色比较

图6-9中，左边的面片使用从六色颜色映射表中获得的索引数据颜色。通过使用这个映射表，顶点的颜色在从蓝绿色过渡到蓝色的过程中仅经过绿色、红色、黄色和洋红色四个过渡色，而右边的面片颜色是均匀过渡的，RGB数值逐步地从一个数值变化到另一个数值，例如，从蓝绿色过渡到绿色会经过以下数值（事实上，每个像素颜色的变化比这里说明的要小得多）：0 1 1, 0 0.9 1, 0 0.8 1, ... 0 0.2 1, 0 0.1 1, 0 0 1。

6.2 定义三维视图

6.2.1 视图概念

视角是指用户指定的、用来显示用户图形或图形场景的特殊方向。视图则是指使用视角变换、切入或切出、改变透视和外观比例等方式显示的一个图形场景。本节的内容主要描述如何定义不同的视图参数。通常的视图都使用三维图形或模型。MATLAB 视图设置由以下两个基本内容组成：

- 设置视点：包括指定用户图形观察点的方位角和高度角、使用镜头定义场景、使用镜头工具栏和图形函数来控制视图（该部分将决定如何使用 MATLAB 镜头视图模型来合成复杂的场景）、设置视图、设置低级镜头属性、选择视图放映类型（包括投影和透视两种类型）等等；
- 设置外观比例。

下面将分别介绍这两个内容。

6.2.2 设置视点

MATLAB 允许用户控制图形在坐标轴中的显示方向。用户可以在图形窗口中指定视点、观察目标、方向和视图的范围。这些视觉特征的控制是通过图形属性集来实现的，用户可以

直接指定这些属性值，也可以使用 view 命令选择一个观察方向，然后 MATLAB 根据这个方向自动进行属性设置，从而定义一个合理的视图。view 命令以坐标轴原点为基准，通过指定方位角和高度角来定义视点。方位角是指视点在 x-y 平面中的极角，一个正的方位角表示视点要从原点处向逆时针方向旋转；高度角则表明该方位角的位置是在 x-y 平面的上方还是下方。

MATLAB 可以根据图形的维数自动选择视点：对于一个二维图形，缺省方位角是 0°，高度角是 90°；对于三维图形，缺省方位角是-37.5°，高度角是 30°。

例如，以下语句创建一个三维曲面图形并在缺省的三维视图中显示这个图形：

```
[X,Y] = meshgrid([-2:.25:2]);
Z = X.*exp(-X.^2 -Y.^2);
surf(X,Y,Z)
```

显示结果如图 6-10 所示。

使用以下语句改变视点设置：

```
view([180 0])
```

于是用户可以从 y 轴的负方向上以高度角 0 来观察这个图形（如图 6-11 所示）。

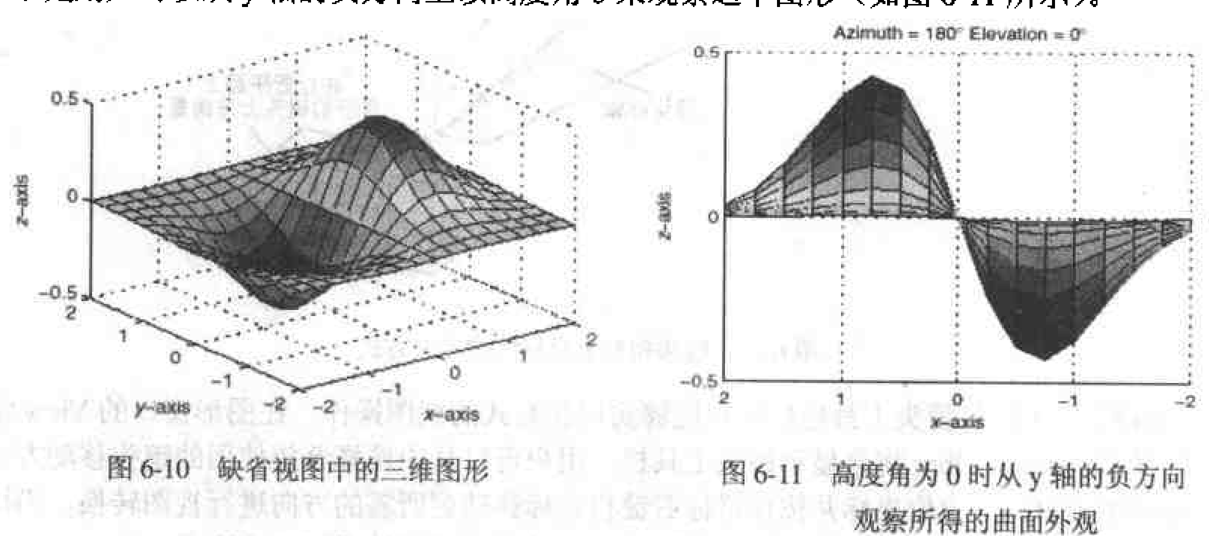


图 6-10 缺省视图中的三维图形　　图 6-11 高度角为 0 时从 y 轴的负方向观察所得的曲面外观

还可以使用一个负的高度角将视点移到坐标轴的下方（观察结果如图 6-12 所示）：

```
view([-37.5 -30])
```

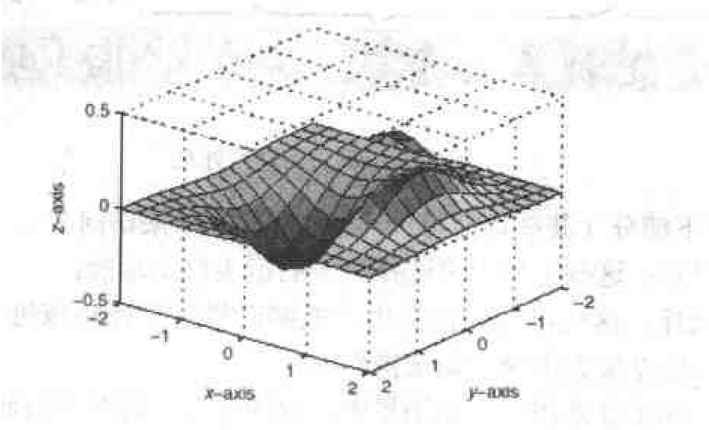


图 6-12 视点在坐标轴下方时观察到的图形

从理论上说，指定视点的方位角和高度角是一种非常简单的操作，但实际上这种指定方式的使用是有限的。用户不能指定视点的准确位置，只能指定视点的方向，观察到的图形结果总是特别强调 z 轴的情况。

1. 镜头控制

除了简单的方位角和高度角调节，MATLAB 镜头提供了更好的图形控制方法。用户观察每一个坐标轴中显示的图形都是在某个特殊空间位置（针对场景的一个特殊方向）处进行的。MATLAB 提供类似于变焦镜头的功能，使用户可以控制 MATLAB 创建的场景视图。图 6-13 说明了镜头是怎样根据坐标轴属性来定义的。

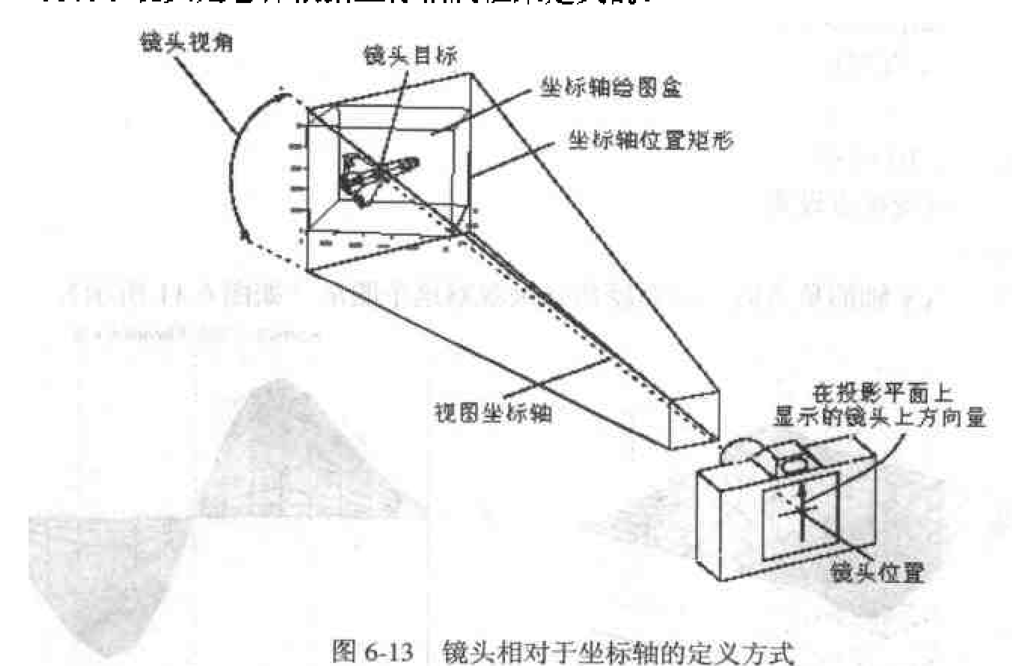


图 6-13　镜头相对于坐标轴的定义方式

MATLAB 提供镜头工具栏使用户能够实现交互式的视图操作。在图形窗口的 View 菜单下选择 Camera Toolbar 将会显示镜头工具栏，用户可以从中选择希望使用的镜头移动方式，通过在图形窗口上定位光标并按住鼠标右键将光标移动到所需的方向进行视图转换。当用户移动鼠标时，MATLAB 实时地刷新视图。镜头工具栏的外观如图 6-14 所示。

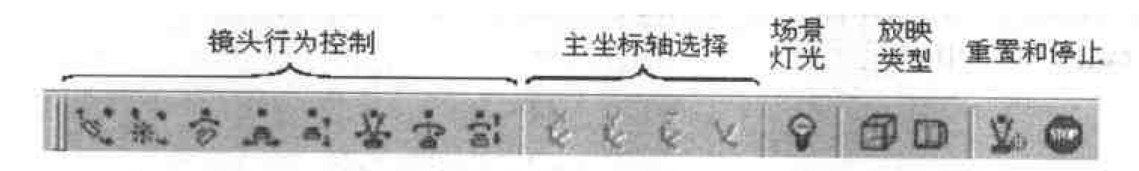


图 6-14　镜头工具栏外观

工具栏包含以下部分（某些功能可以通过 Tools 菜单来访问）：

- 镜头行为控制：这些工具用来选择可用的镜头移动功能；
- 主坐标轴选择：这些可选项使得用户能够选择主坐标轴或进行无轴操作（注意某些镜头控制方法是根据主坐标轴来操作的）；
- 场景灯光：场景灯光按钮控制场景中光源的开关（每个坐标轴对应一个光源）；
- 放映方式：可以选择投影或透视放映类型；
- 重置和停止：重置功能能够使场景返回到标准的三维视图中，而停止将使得镜头停

止移动（如果用户使光标移动过于频繁，这个操作将非常有用）。

首先介绍镜头行为控制方法。镜头行为控制主要讨论单个镜头行为控制功能，共有8个镜头行为控制工具：

- 轨道镜头：缺省情况下，轨道镜头按照 z 轴旋转镜头。用户可以通过设置主坐标轴来任意选择旋转轴。未定义主坐标轴时，用户可以按照任意轴对图形进行旋转。当保持镜头目标固定时，轨道镜头将改变镜头的位置属性；
- 场景灯光：根据镜头位置放置一个光源。缺省情况下，场景灯光位于镜头的右边。轨道场景灯光将改变光源与镜头位置的差异。虽然 MATLAB 只提供一个场景灯光，但用户可以通过 light 命令添加其他灯光。通过工具栏的黄色灯泡图标来控制灯光的开关。轨道场景灯光通过改变灯光的位置属性来移动场景灯光；
- 持平/翻转镜头：在保持镜头固定的同时移动镜头，使其指向场景中的某个点。缺省情况下，这种行为是按照环绕 z 轴的一条弧线进行的，用户可以通过选择主坐标轴来任意指定转轴。这些功能通过改变镜头目标属性来实现镜头指向点的变化；
- 水平/垂直移动镜头：使场景在鼠标移动时按照水平或垂直方向移动。这是通过同时改变镜头位置和目标属性实现的；
- 向前或向后移动镜头：鼠标向上或向左移动将使镜头逼近场景，鼠标向下或向左移动则将使镜头远离场景。允许镜头穿过某一场景中的物体移动到镜头目标的另一边。镜头位置的移动是沿着镜头位置和镜头目标间的连线来进行的。移动镜头有点类似于在驾车过程中驾驶员的目光始终保持向前。当用户向右转弯时，场景中的物体向左移动，当视图坐标轴位于一个垂直于主坐标轴的平面时移动镜头效果最为明显。移动镜头同时改变镜头位置和镜头目标并保持两者之间的距离不变；
- 放大镜头：如果鼠标向上或向右移动时，场景将会变大，否则变小。镜头放大并不移动镜头，因此不会使视点穿越物体。放大是通过改变镜头视角来实现的，视角越大，场景显得越小；
- 滚动镜头：滚动镜头根据视图的坐标轴旋转镜头从而旋转视图。滚动镜头改变镜头上向量的属性；
- 移动镜头：将镜头沿镜头目标方向移动，同时镜头目标也跟着移动。移动镜头使得镜头能够穿越场景。

场景的主坐标轴用于定义视图上方的导向。例如，MATLAB 曲面是将 z 轴的正方向作为主坐标轴来排列图形的。主坐标轴限制镜头沿着与主坐标轴平行或垂直的坐标轴方向移动。如果用户数据是由一个特定的坐标轴定义的，那么指定一个主坐标轴将非常有用。在视图中，旋转轴是通过一条垂直线和一条水平线决定的，这两条线均通过由镜头目标属性定义的点与主坐标轴平行或正交。这就意味着场景或镜头将按照一条中心在镜头目标处的曲线移动。图 6-15 表明了 z 轴作为主坐标轴时的旋转轴。

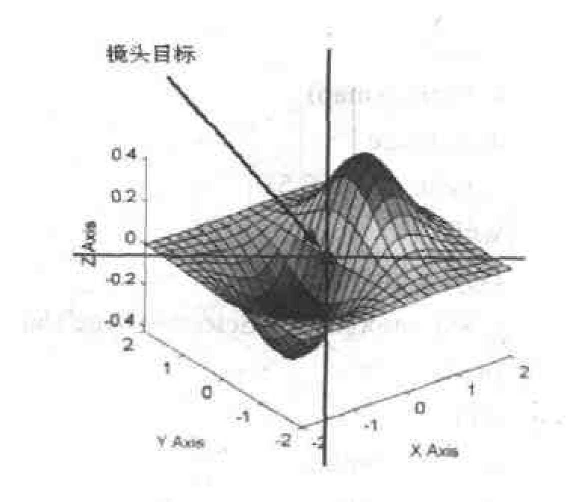


图 6-15 z 轴作为主坐标轴时的旋转轴

在图 6-15 中，鼠标水平移动将导致曲面按

竖置轴旋转，而鼠标竖直移动将导致曲面按照水平轴旋转。

有三个镜头行为控制工具可以用来指定主坐标轴：轨道镜头、持平/翻转镜头和移动镜头。这里要注意，当用户创建图形时，MATLAB 使用与图形窗口相匹配的外观比例显示图形，这种比例不一定能够为三维图形操作生成一个最优状态，因而有时镜头在场景中的旋转可能会导致图形扭曲。为了避免这种问题，最好使用三维可视化模型（在命令行中使用 axisvis3d 命令获得）。

2. 镜头函数

使用镜头函数也能够控制镜头的行为。表 6-2 列出了用户可以用来实现镜头变换的 MATLAB 函数及其功能。

表 6-2　实现镜头变换的 MATLAB 函数及其功能

函数名	功　能
Camdolly	移动镜头位置和目标
camlookat	观察指定对象
camorbit	根据镜头目标指定镜头轨道
campan	根据镜头位置旋转镜头目标
campos	设置或获得镜头位置
camproj	设置或获得放映类型（投影或透视）
camroll	根据视图坐标轴旋转镜头
camtarget	设置或获得镜头目标位置
camup	设置或获得镜头上向量数值
camva	设置或获得镜头视角数值
camzoom	放大或缩小场景

使用两个例子来说明这些函数的使用方法。

首先说明如何实现镜头的移动。首先装载一幅图像，使用 ginput 函数获得图像的坐标位置，然后根据该坐标利用 camdolly 函数的数据坐标选项将镜头和镜头目标移动到新的位置，最后使用 camva 函数放大视图并固定镜头视角（否则视角将由 MATLAB 自动控制）。程序代码如下：

```
load cape
image(X)
colormap(map)
axis image
camva(camva/2.5)
while 1
[x,y] = ginput(1);
if ~strcmp(get(gcf,'SelectionType'),'normal')
break
end
ct = camtarget;
dx = x - ct(1);
```

```
    dy = y - ct(2);
    camdolly(dx,dy,ct(3),'movetarget','data')
    drawnow
    end
```

下面给出一幅描述气流穿越场景的图例。首先使用函数 isosurface、isonormals、reducepatch 和 coneplot 绘制曲面以及气流描述锥体。将数据的外观比例（daspect）设置为[1,1,1]以保证 MATLAB 能够正确地计算最终视图中锥体的大小。程序代码如下：

```
load wind
wind_speed = sqrt(u.^2 + v.^2 + w.^2);
hpatch = patch(isosurface(x,y,z,wind_speed,35));
isonormals(x,y,z,wind_speed,hpatch)
set(hpatch,'FaceColor','red','EdgeColor','none');
[f vt] = reducepatch(isosurface(x,y,z,wind_speed,45),0.05);
daspect([1,1,1]);
hcone = coneplot(x,y,z,u,v,w,vt(:,1),vt(:,2),vt(:,3),2);
set(hcone,'FaceColor','blue','EdgeColor','none');
```

选择透视放映方式来观察镜头穿过曲面的深度，设置镜头视角为固定值：

```
camproj perspective
camva(25)
```

在镜头位置处创建光源，即提供一个与镜头一起穿过曲面的光源。设置曲面的反射属性（较暗，AmbientStrength=0.1）和锥体的反射属性（亮，SpecularStrength 和 DiffuseStrength 为 1）：

```
hlight = camlight('headlight');
set(hpatch,'AmbientStrength',.1,...
'SpecularStrength',1,...
'DiffuseStrength',1);
set(hcone,'SpecularStrength',1);
set(gcf,'Color','k')
```

由于本例中使用了灯光，所以 MATLAB 必须选择 zbuffer 着色方式。另外，如果用户拥有 OpenGL 库，MATLAB 也可以使用 OpenGL 的着色方法。

```
lighting phong
set(gcf,'Renderer','zbuffer')
```

定义镜头路径为流线型：从点(80,309,11)处开始创建流线，获取该流线每一点的坐标值，然后将该流线的图形删除（这是因为只需要流线的数据而不需要将该流线真正绘制出来）。

```
hsline = streamline(x,y,z,u,v,w,80,30,11);
xd = get(hsline,'XData');
yd = get(hsline,'YData');
zd = get(hsline,'ZData');
delete(hsline)
```

实现穿越：重画视图使镜头位置和目标根据流线坐标移动，保证镜头和灯光同时移动并

显示每一次移动的结果。

```
for i=1:length(xd)-50
campos([xd(i),yd(i),zd(i)])
camtarget([xd(i+5)+min(xd)/100,yd(i),zd(i)])
camlight(hlight,'headlight')
drawnow
end
```

图 6-16 至图 6-18 分别是视图在 i=10，i=110 和 i=185 时的快照。

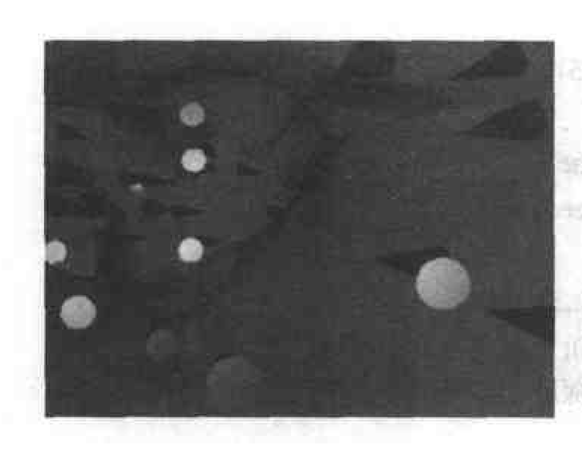


图 6-16　i=10 时的快照

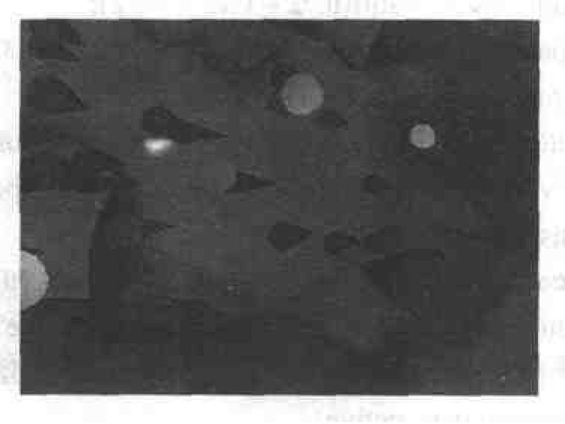


图 6-17　i=110 时的快照

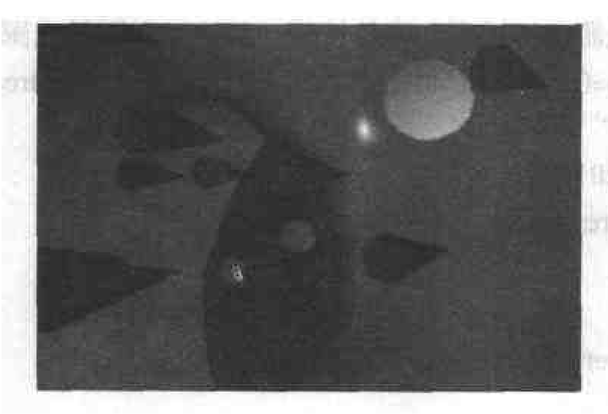


图 6-18　i=185 时的快照

镜头图形是建立在控制镜头位置和方向的坐标轴属性群之上的。通常情况下使用镜头命令并不需要访问这些属性。表 6-3 给出了这些属性的名称、含义及部分缺省值。

表 6-3　控制镜头的坐标轴属性及其含义

属 性 名 称	含　义	缺 省 值
CameraPosition	按照坐标轴单位指定镜头位置	由 MATLAB 指定
CameraPosition Mode	自动模式时由 MATLAB 指定镜头位置，人工模式下由用户指定镜头位置	自动模式
CameraTarget	指定镜头所指的点在坐标轴上的位置，与 CameraPosition 一起定义视图的坐标轴	图形盒中心位置
CameraTargetMode	自动模式时由 MATLAB 指定镜头目标位置，人工模式下由用户指定镜头目标位置	自动模式

（续表）

属 性 名 称	含 义	缺 省 值
CameraUpVector	镜头沿坐标轴的旋转由一个向上的向量来定义	二维图形中该向量与 y 轴的正方向相同，三维图形沿 z 轴的正方向
CameraUpVectorMode	自动模式：MATLAB 指定 人工模式：用户指定	自动模式
CameraViewAngle	指定镜头透镜的观察范围	调节视角为能够捕获整个场景的最小视角
CameraViewAngleMode	自动模式：MATLAB 指定视角 人工模式：用户指定视角	自动模式
Projection	选择投影放映或透视放映	投影放映

使用这些属性对视图进行操作的方法不一而同，下面给出几个常用的例子。

首先说明如何实现镜头逼近或远离目标。为了使镜头沿着视图坐标轴移动，首先要为 CameraPosition 属性计算新的坐标值，同时增大或减小镜头位置和目标之间的距离。函数 movecamera 可以用于计算新的镜头位置，如果该函数的参数 dist 为正，则镜头向场景内移动，否则向场景外移动。

```
function movecamera(dist)
set(gca,'CameraViewAngleMode','manual')
newcp = cpos-dist*(cpos-ctarg);
set(gca,'CameraPosition',newcp)
function out = cpos
out = get(gca,'CameraPosition');
function out = ctarg
out = get(gca,'CameraTarget');
```

下面说明如何使视点在场景中旋转。使用函数 orbit 使镜头在场景中旋转：

```
function orbit(deg)
[az el] = view;
rotvec = 0:deg/10:deg;
for i = 1:length(rotvec)
view([az+rotvec(i) el])
drawnow
end
```

第三个例子是计算镜头上向量。假设要指定上向量位于 y-z 平面与 z 轴成 30° 角处，使用以下表达式：

```
upvec = [cos(90*(pi/180)),cos(60*(pi/180)),cos(30*(pi/180))];
set(gca,'CameraUpVector',upvec)
```

以上语句将产生如图 6-19 所示的图形。

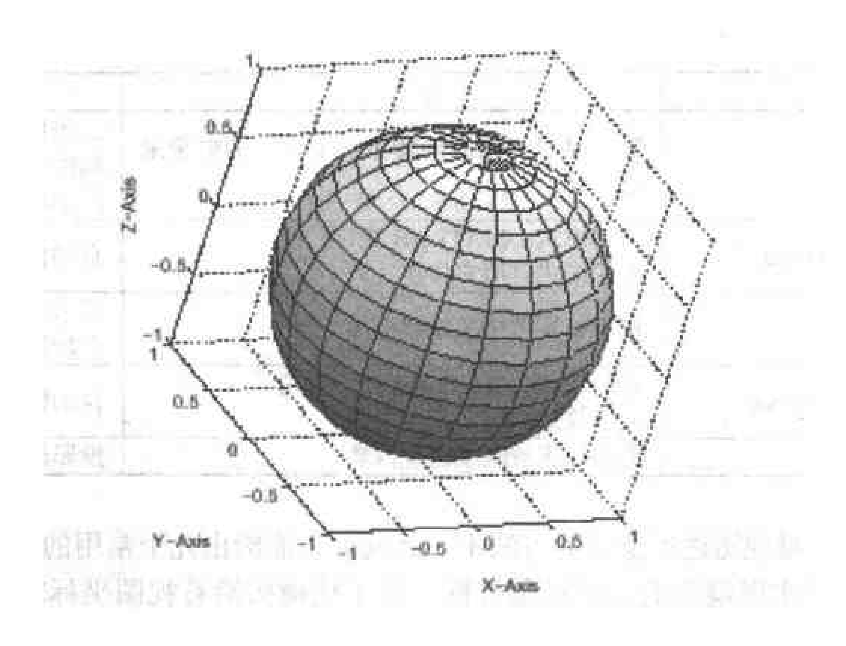


图 6-19　上向量位于 y-z 平面上与 z 轴成 30° 角时的图形

3. 着色与放映方式

MATLAB 支持投影和透视两种三维图形放映方式，放映类型的选择依赖于待显示图形的类型。投影模式将视图流作为正平行六面体来放映，物体与镜头的距离并不影响物体的大小。如果希望保持物体的真实大小以及物体间所成的角度，这种放映方式比较实用。透视模式将视图流作为平顶金字塔来放映，此时物体与镜头的距离将会导致透视距离变化，使物体外观发生一定的变化。当用户希望体现实际物体的真实感时最好使用这种方式。

缺省情况下，MATLAB 通过调节镜头位置、镜头目标、镜头视角等属性使镜头位于场景的中心位置，并使得用户可以通过镜头看到坐标轴中的所有物体。如果用户改变镜头位置使某些图形对象位于镜头的后面，那么坐标轴的放映方式属性和图形着色方法会影响场景的显示效果。表 6-4 说明了放映类型和着色方法的交互作用。

表 6-4　放映类型和着色方法的交互作用

放映方式 / 着色方式	投　影	透　视
zbuffer	视角将决定目标处的场景范围	视角决定从镜头位置到无穷远处的场景范围
Painter	无论镜头位置在何处均显示所有物体	如果图形对象在镜头后面，这种方式不被推荐

6.2.3　设置外观比例

坐标轴通过设置坐标轴的比例尺和刻度范围来刻画图形对象的外观。在用户创建图形时，MATLAB 根据待绘制数据的大小或数值自动确定坐标轴的比例尺，然后根据可获得显示空间的大小来绘制坐标轴。坐标轴外观比例属性将控制 MATLAB 如何选取创建图形时所需的坐标轴比例。

缺省情况下，MATLAB 绘图使用的坐标轴尺寸比图形窗口尺寸略小，使图形窗口留有

边框。如果用户改变了图形窗口的尺寸，坐标轴的尺寸和x，y轴的比例也会相应变化，这就使坐标轴总是能够填满窗口中的可获得空间。MATLAB还设置x，y和z轴的范围从而为每一个方向提供最大限度的显示空间。有时用户可能希望图形总是按照真实尺寸来显示，或者希望图形在打印对话框中以固定的大小显示（预览），此时就需要用户自己来控制坐标轴的拉伸。

axis命令可以用来调节图形的比例。缺省情况下，MATLAB搜索待绘制数据的最大最小值从而选择相应的坐标轴范围，用户也可以通过设置坐标轴范围覆盖缺省的坐标轴范围。

axis([xmin xmax ymin ymax zmin zmax])

用户可以使用预定义的axis选项来控制MATLAB刻画坐标轴的方法：

- axis auto：返回坐标轴的缺省刻度；
- axis manual：冻结当前坐标轴范围的刻度（如果用户设置了hold on，那么绘图结果仍使用当前的范围）；
- axis tight：设置坐标轴范围为数据的范围；
- axis ij：使MATLAB进入矩阵坐标轴模式：坐标轴系统的原点在左上角，i轴的方向是垂直从上到下的，j轴是水平从左到右的；
- axis xy：使MATLAB进入缺省的笛卡尔坐标模式：坐标轴系统的原点在左下角，x轴的方向是水平从左到右，y轴的方向是垂直从下往上的。

axis命令还能使用户调节图形的显示比例。通常MATLAB会自动拉伸坐标轴以适应整个窗口，但在许多种情况下，需要根据图形的特征来定义坐标轴的外观比例。axis命令提供许多用来调节外观比例的选项，例如：

- axis equal：改变当前的坐标刻度使得坐标轴x，y和z的等值刻度以相同的长度增长（即比例尺相同）；
- axis square：使每个坐标轴等长并覆盖自动拉伸行为；
- axis vis3d：冻结外观比例属性并覆盖自动拉伸行为。在用户旋转场景时，该选项能够保证各项设置不发生变化；
- axis image：使得坐标轴的外观比例与原图形一致；
- axis normal：将当前的坐标轴恢复为最大尺寸并删除单位刻度限制。

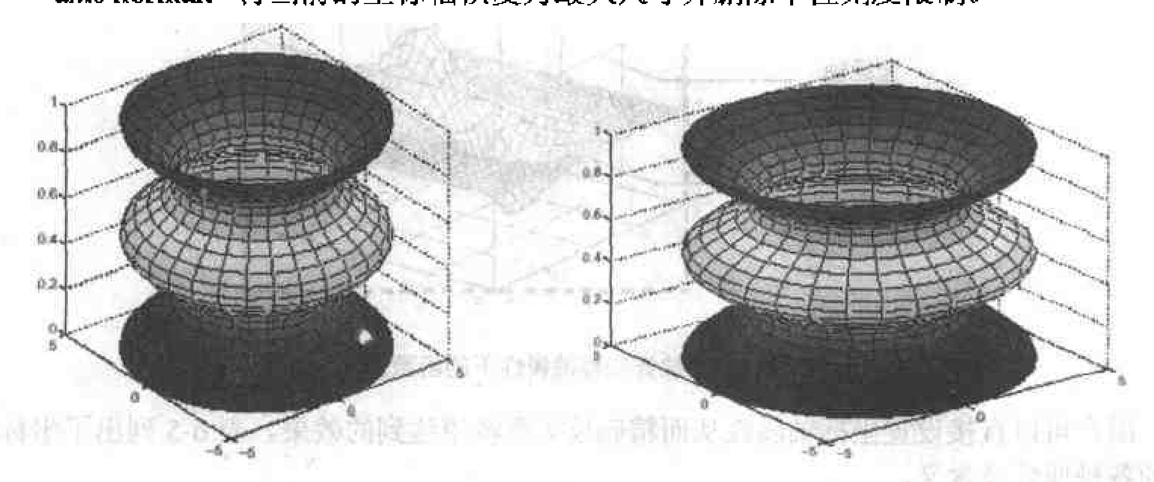


图6-20 axis normal产生的图形效果　　图6-21 axis square产生的图形效果

图6-20至图6-22给出了三种不同坐标轴选项对以下语句生成柱面所产生的影响：

```
t = 0:pi/6:4*pi;
[x,y,z] = cylinder(4+cos(t),30);
surf(x,y,z)
```

axis normal 是缺省选项，MATLAB 根据数据范围自动设置坐标轴的范围并拉伸图形以适应整个图形窗口。

axis square 将产生与图形窗口形状无关的正方形坐标系，因而柱面不会被拉伸。但此时 x 和 y 轴的刻度以 5 为单位增长，而 z 轴则以 0.1 为单位增长。

axis equal 使得每个坐标轴的数据单位都相等，柱面也不会被拉伸。

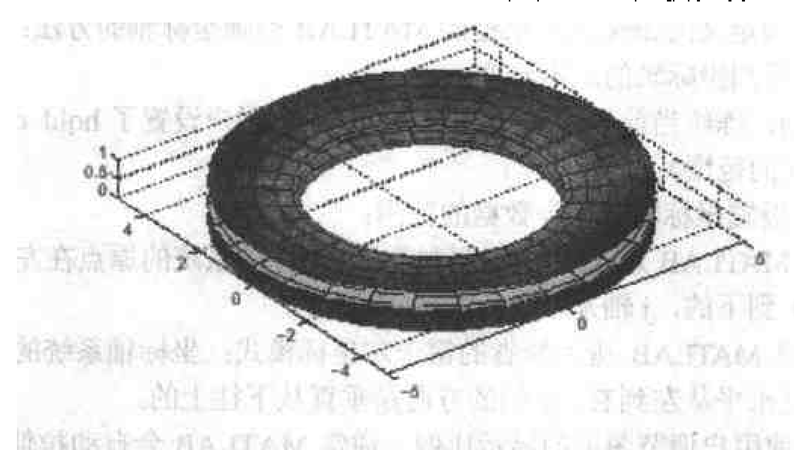


图 6-22　axis equal 产生的图形效果

axis 是通过设置坐标轴对象的属性来工作的，例如对于函数 $z=x*\exp(-x^2-y^2)$（$-2\leqslant x\leqslant 2$，$-4\leqslant y\leqslant 4$），使用 MATLAB 缺省坐标轴属性绘制该函数的图形：

```
[X,Y] = meshgrid([-2:.15:2],[-4:.3:4]);
Z = X.*exp(-X.^2 - Y.^2);
mesh(X,Y,Z)
```

绘图结果如图 6-23 所示。

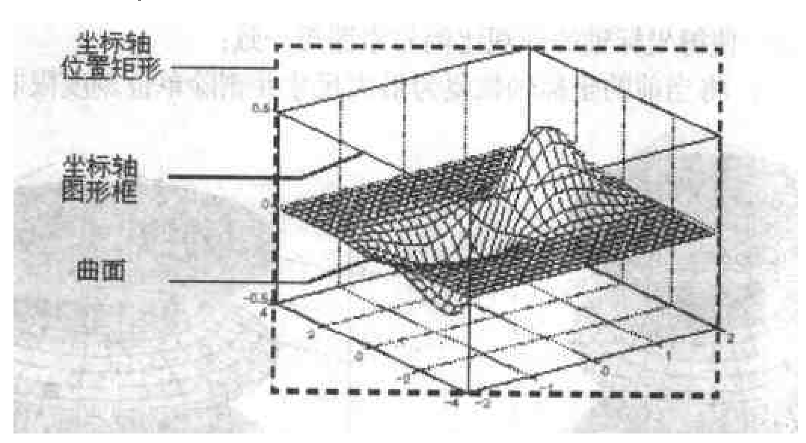


图 6-23　缺省坐标轴属性下的函数图形

用户可以直接设置坐标轴属性从而精确地实现希望达到的效果。表 6-5 列出了坐标轴对象的各种属性及含义。

表 6-5 坐标轴对象属性及含义

属 性 名 称	含 义
DataAspectRatio	设置各坐标轴的相对刻度，[1，1，1]表示真实比例
DataAspectRatioMode	在自动模式中，MATLAB 提供可获得空间中最大的坐标轴刻度
PlotBoxAspectRatio	设置坐标轴图形框的比例
PlotBoxAspectRatioMode	自动模式中，MATLAB 将设置 PlotBoxAspectRatio 为[1，1，1]，除非用户精确定义了 DataAspectRatio 或坐标轴范围
Position	使用一个四元素向量[left offset, bottom offset, width, height]定义坐标轴的位置和尺寸
XLim, YLim, ZLim	设置各坐标轴的最大最小值
XLimMode, YLimMode,ZLimMode	在自动模式中，由 MATLAB 选择坐标轴的最大最小值

例如，以下语句定义了一个楔型片状物体：

patch('Vertices',vertex_list,'Faces',vertex_connection,...

'FaceColor','w','EdgeColor','k')

view(3)

vertex_list =

0 0 0；0 1 0；1 1 0；1 0 0；0 0 1；0 1 1；1 1 4；1 0 4

vertex_connection =

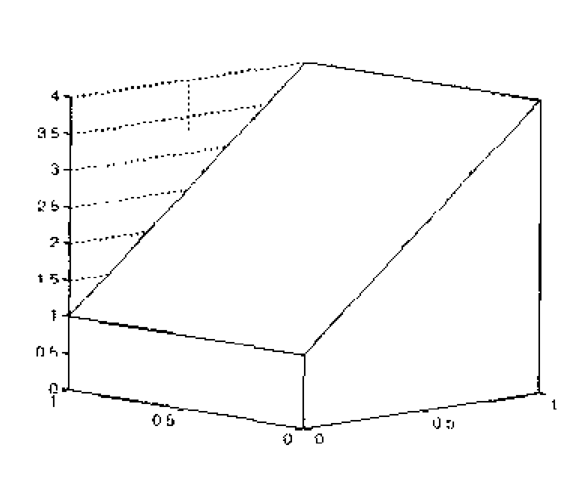
1 2 3 4；2 6 7 3；4 3 7 8；1 5 8 4；1 2 6 5；5 6 7 8

使用缺省的属性显示该物体的结果如图 6-24 所示。

显然，这个图形扭曲了这些数据所给的真实物体的形状。为了显示正确的比例，使用以下语句将 DataAspectRatio 属性设置为真实比例[1,1,1]：

set(gca,'DataAspectRatio',[1 1 1])

再调用显示函数就可以得到如图 6-25 所示的真实的物体外形。



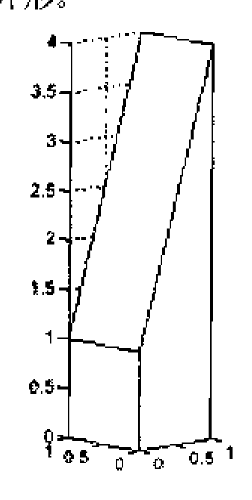



图 6-24 改变缺省设置后所得的楔状图形

图 6-25 楔状图形的真实外观

用户还可以通过三种其他方式来控制图形的外观比例：

- 指定 x，y 和 z 轴的相对刻度（数据外观比例）；
- 指定由坐标轴定义的空间形状；
- 指定坐标轴范围。

这三种方法是通过以下命令实现的：

daspect：设置或查询数据外观比例；

pbaspect：设置或查询图形盒的外观比例；

xlim：设置或查询 x 轴的范围；

ylim：设置或查询 y 轴的范围；

zlim：设置或查询 z 轴的范围。

6.3 三维对象的灯光渲染及透明处理

6.3.1 基本概念

灯光是一种能够为图形场景增加真实感的可视化技术。MATLAB 通过模拟物体在自然光源下的加亮区域和暗淡区域来实现灯光效果。为了制造灯光效果，MATLAB 定义了一种名为“灯光”的图形对象。

MATLAB 提供一些灯光命令使用户能够设置光源位置并调节被照亮物体的特征。直接使用灯光或被照亮物体的属性设置命令以获得特定的效果通常是非常有效的，这些命令包括：

camlight：根据镜头位置创建或移动光源；

lightangle：在球坐标中创建或移动光源；

light：创建灯光对象；

lighting：选择照亮方法；

material：设置被照亮物体的反射属性。

用户可以使用 light 函数创建一个灯光对象。这里要注意，通常创建一个灯光将会引起一些与灯光有关的控制特征的变化，例如周围环境灯光或物体的反射属性等。灯光对象有三种重要的属性：

color：投向被照亮物体的光源色彩；

style：照明距离可以为无穷远（缺省）也可以为局部；

position：无穷远灯光的方向或局部光源的位置

下面给出一个例子来说明如何在场景中添加灯光。以下语句将显示一个薄膜曲面，该曲面将被一个由位置向量[0 –2 1]指定的光源照亮。

```
membrane
light('Position',[0 -2 1])
```

绘制结果如图 6-26 所示。

灯光还可以给数学函数产生的曲面图形照明，例如以下函数：

```
ezsurf('sin(sqrt(x^2+y^2))/sqrt(x^2+y^2)',[-6*pi,6*pi])
```

无灯光时的曲面外观如图 6-27 所示。

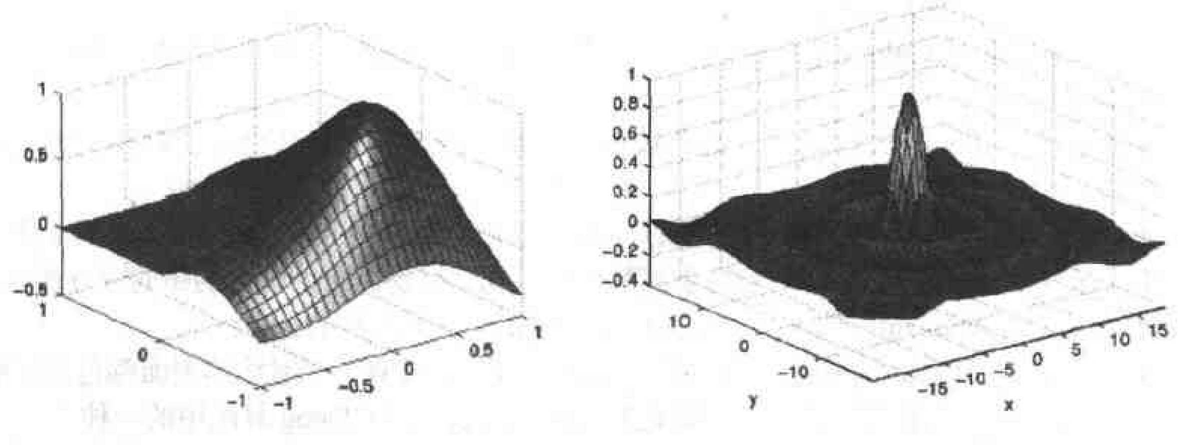


图 6-26 定向光源照亮的薄膜曲面　　图 6-27 无光照的函数曲面外观

使用以下语句为该函数曲面添加灯光：

```
view(0,75)
shading interp
lightangle(-45,30)
set(gcf,'Renderer','zbuffer')
set(findobj(gca,'type','surface'),'FaceLighting','phong',...
'AmbientStrength',.3,'DiffuseStrength',.8,
'SpecularStrength',.9,'SpecularExponent',25,...
'BackFaceLighting','unlit')
```

光照效果如图 6-28 所示。

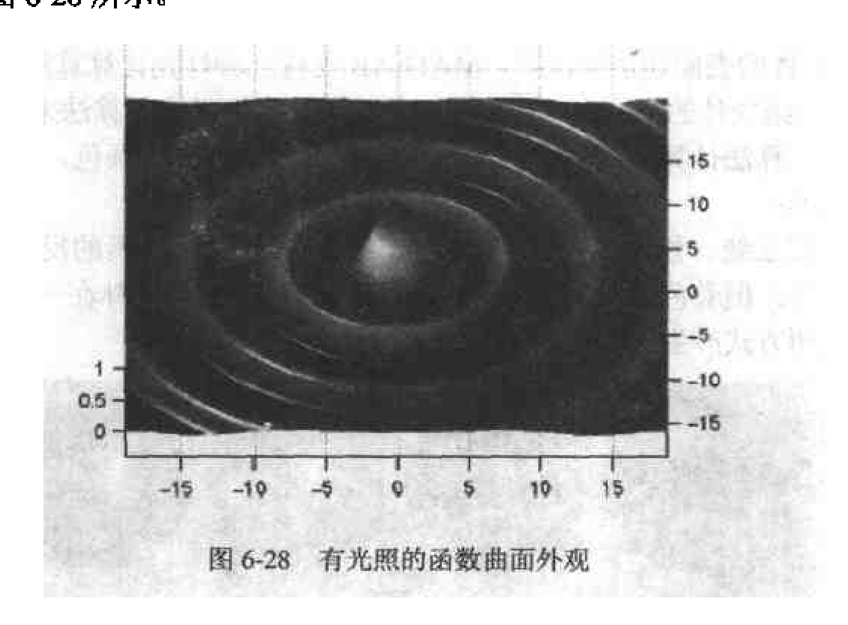


图 6-28 有光照的函数曲面外观

6.3.2 灯光对象及其属性

灯光对象本身是不可见的，但是灯光对任何平面或曲面的影响都是可见的。用户可以通过设置不同的坐标值、灯光、面片或曲面对象属性来改变灯光效果。

常用的灯光属性有以下几种：

- AmbientLightColor：指定场景背光灯颜色的坐标轴属性，该属性对不同物体的影响不同；
- AmbientStrength：这是一个面片或曲面属性，该属性决定从物体反射回来的周围环境分量的光照强度；
- DiffuseStrength：该面片或曲面属性决定从物体反射回来的灯光的散射成分的强度；
- SpecularStrength：该面片或曲面属性决定从物体反射回来的灯光的镜向成分的强度；
- SpecularExponent：该面片或曲面属性决定镜向加亮区域的大小；
- SpecularColorReflectance：该面片或曲面属性决定计算灯光对物体表面颜色的影响的算法，可以选择无灯光，也可以选择 flat、Gouraud 和 Phong 算法中的一种；
- EdgeLighting：该面片或曲面属性决定灯光对物体边缘影响的计算算法；
- BackFaceLighting：该面片或曲面属性决定曲面远离镜头时的照明方式。这对于区分物体的内部和外部非常有用；
- FaceColor：该面片或曲面属性决定对象表面的颜色；
- EdgeColor：该面片或曲面属性决定对象边缘的颜色；
- VertexNormal：该面片或曲面属性给对象的每个顶点包含一个标准向量，MATLAB 使用这个向量来实现灯光的计算；
- NormalMode：该面片或曲面属性决定当用户改变对象数据值或 VertexNormal 属性值时，MATLAB 是否重新计算顶点向量。

下面将一一介绍这些属性的作用和效果。

照明方法：当用户为一个坐标轴添加一个灯光时，由 MATLAB 来决定该轴上被显示平面或曲面所受的灯光影响。可以选择不使用任何照明方式，也可以选择不同的灯光计算算法来计算被照亮物体的表面和边缘色彩。MATLAB 支持三种灯光计算算法：

- Flat 算法给物体的每个表面生成相同的颜色。多选择这种算法来显示多面体；
- Gouraud 算法计算每个顶点的颜色，然后据此修改表面颜色。多选择这种算法观察曲面图形；
- Phong 算法统一修改每个表面的顶点，然后计算每个像素的反射程度。这种算法的效果较好，但着色时间较长。图 6-29 给出了一个红色球体在一个白色光源下的四种不同照明方式产生的效果。

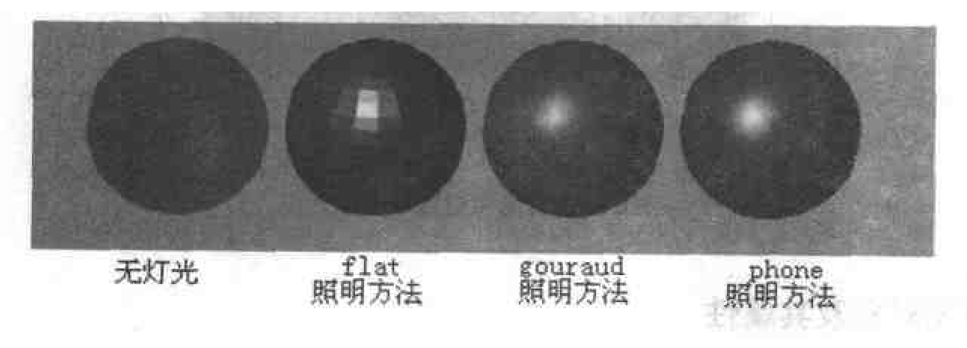


图 6-29　红色球体白色光源下三种照明方式的不同效果

图形对象的反射特征：用户可以修改平面和曲面图形对象的反射特征，从而改变对象在有灯光场景中的外观。这些特征包括镜向反射和散射、周围环境灯光、镜向指数、镜向颜色

反射、背光、镜向反射和散射。用户可以通过修改 SpecularStrength 和 DiffuseStrength 属性来控制物体表面的镜向反射和散射程度。图 6-30 说明了不同设置下的不同效果。

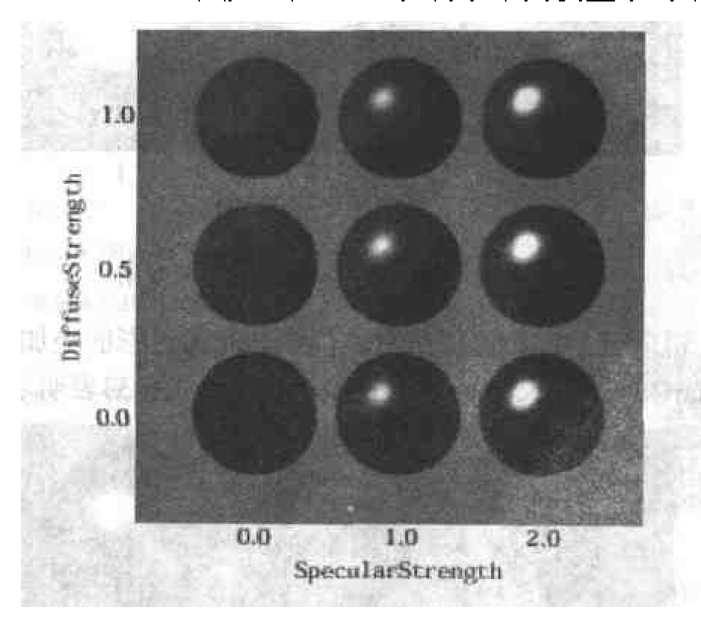


图 6-30 不同反射特征下的显示效果

环境灯光：环境灯光是一种无方向的、统一照亮场景中所有物体的灯光。仅仅当坐标轴中存在灯光对象时，环境灯光的效果才是可见的。有两个属性可以用来控制环境灯光的行为：AmbientLightColor和AmbientStrength。图6-31表示了一个红球在白光源下通过设置不同属性产生的不同效果。

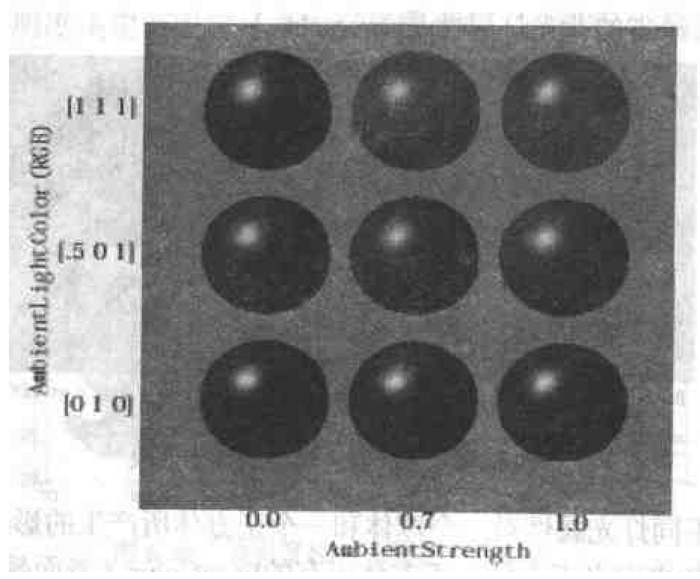


图 6-31 不同环境灯光下的显示效果

镜向指数：镜向加亮区的大小依赖于平面和曲面对象的SpecularExponent属性。该属性的取值范围是1到500，通常取5到20即可。

图6-32显示了该属性对红球白光源场景的影响。

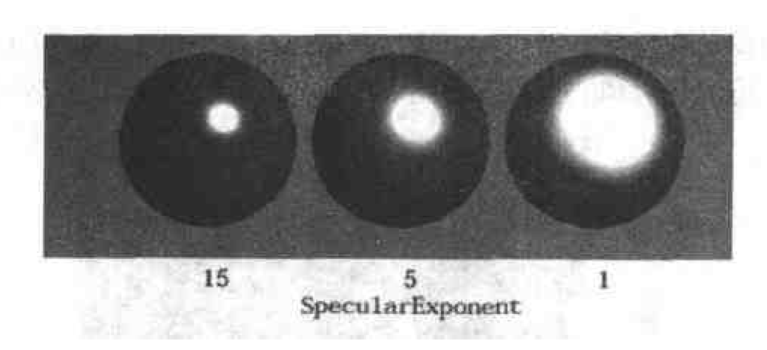


图6-32　不同镜向指数下的显示效果

镜向色彩反射：镜向反射色彩可以从对象色彩与光源色彩的叠加（0）到单光源色彩（1）逐渐变化。由SpecularColorReflectance属性控制该色彩。图6-33表明了该属性的影响。

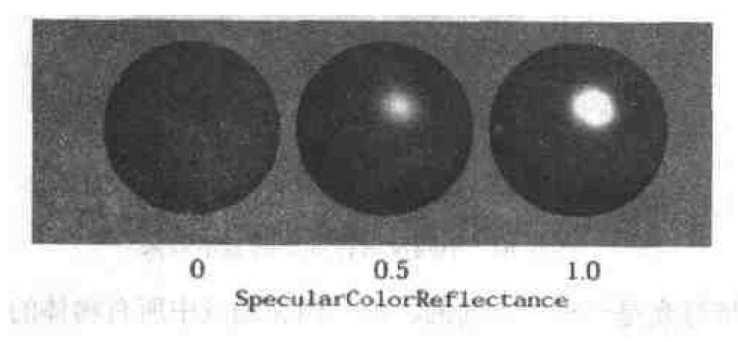


图6-33　不同镜向色彩反射下的显示效果

背光灯：背光灯在区分物体的内部和外部时非常有用。图6-34的柱型切面说明了背光灯对物体外观的影响（缺省的背光灯属性是reverselit。）。

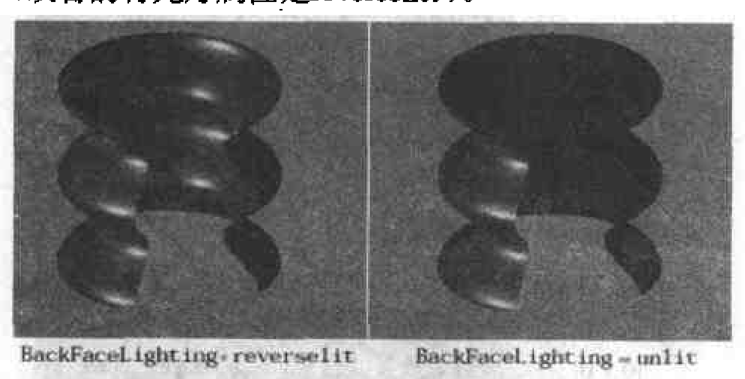


图6-34　不同背光灯下的显示效果

下面举例说明不同灯光属性对一个球体和一个正方体所产生的影响。使用变量vert和fac作为patch函数的参数来定义正方体，正方体所有的FaceColor（表面颜色）属性都设置为黄色。函数sphere创建一个球形曲面，该曲面的句柄由findobj函数获得并用来搜索type属性为surface的对象。light函数定义了两个位于无穷远处、角度由Position向量确定的白色光源。使用flat照明方式来提高正方体每个面的可视性。球体曲面使用Phong照明方式产生平滑的灯光效果。material shiny命令同时影响正方体和球体的反射属性（球体反射效果较为明显）。由于球体是封闭的，所以对BackFace- Light属性作了修改，将原来远离镜头的面的顶点从统一反色变为正常显示，消除了不必要的边缘效果。

使用的语句如下：

```
vert =
1 1 1; 1 2 1; 2 2 1; 2 1 1; 1 1 2; 1 2 2; 2 2 2; 2 1 2;
fac =
1 2 3 4; 2 6 7 3; 4 3 7 8; 1 5 8 4; 1 2 6 5; 5 6 7 8
sphere(36);
h = findobj('Type','surface');
set(h,'FaceLighting','phong','FaceColor','interp',...
'EdgeColor',[.4 .4 .4], 'BackFaceLighting','lit')
hold on
patch('faces',fac,'vertices',vert,'FaceColor','y');
light('Position',[1 3 2]);
light('Position',[-3 -1 3]);
material shiny
axis vis3d off
hold off
```

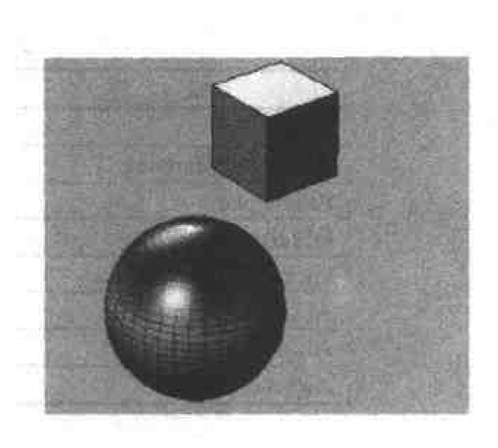


图 6-35 不同灯光属性下的显示效果

显示效果如图6-35所示。

用户可以通过查看lighting和material M文件的源代码来理解不同属性对灯光的影响。

6.3.3 物体透明化

一个图形对象的透明度决定着用户目光能够穿透该物体的程度。用户可以将物体的透明度指定为完全透明到完全不透明范围内的任意透明程度。支持透明的图形对象有：图像、面片和曲面。图6-36说明了透明度对图形的影响，可以看出，图中的绿色曲面内部隐藏着多个小锥形体。

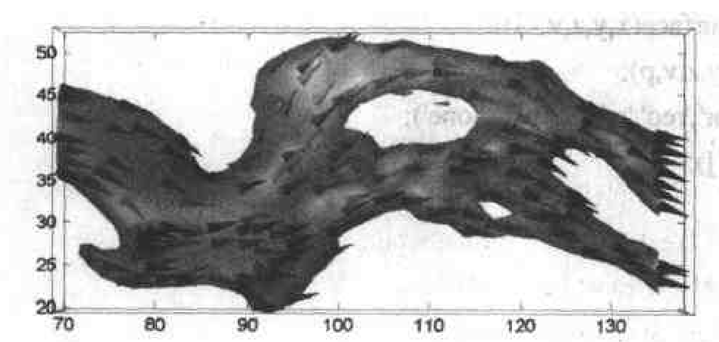


图 6-36 透明度显示隐藏物体的效果

注意，如果需要使用透明度，用户必须拥有OpenGL库，MATLAB会自动调用OpenGL来实现透明度操作。

6.3.4 设置透明度数值

透明度的数值通常称为α值，其变化范围是[0,1]。α值为0表示完全透明（即不可见），

α值为1表示完全不透明。MATLAB对透明度的处理方法与其处理单个物体颜色的方法类似：

面片和曲面既能够定义一个单表面和边缘α值，也可以根据图形α映射表的数值实现统一或插值透明度。使用透明度属性可以精确地设置物体的透明度，表6-6列出了一些常用的透明度属性。

表6-6　常用透明度属性及含义

属　性	含　义
AlphaData	图像和曲面对象的透明度数据
AlphaDataMapping	透明度数据映射方式
FaceAlpha	表面透明度（面片和曲面独有属性）
EdgeAlpha	边缘透明度（面片和曲面独有属性）
FaceVertexAlphaData	α值属性（面片独有）
ALim	α坐标轴范围
ALimMode	α坐标轴范围模式
Alphamap	图形窗口α映射表

有3个简单函数可以用来设置α值的属性：

alpha：设置或查询当前坐标轴物体的透明度属性；

alphamap：指定图形窗口的α映射表；

alim：设置或查询坐标轴的α值范围。

如果用户希望体现被不透明物体遮盖的结构，那么为图形对象指定一个单透明度数值是非常有效的，可以使用FaceAlpha和EdgeAlpha属性指定面片和曲面表面和边缘的透明度。下面的例子使用flow函数生成一个喷气机在无限大水槽中运动的速度特性数据，然后使这些数据可视化。采用的可视化方法是创建一个透明度不同的曲面来说明不同位置处的水流速度。

首先创建速度特性数据并采用不透明曲面描述这些数据：

```
[x y z v] = flow;
p = patch(isosurface(x,y,z,v,-3));
isonormals(x,y,z,v,p);
set(p,'facecolor','red','edgecolor','none');
daspect([1 1 1]);
view(3);
axis tight;
grid on;
camlight;
lighting gouraud;
```

图形结果如图6-37所示。从该图中不能得到喷气机对水流影响的详细情况。

通过给曲面添加透明度可以说明实际的水流情况比使用不透明曲面所能看到的水流情况要复杂的多。设置曲面的FaceAlpha（表面透明度）属性为0.5：

```
alpha(.5)
```

绘图结果如图6-38所示。显然，从该图中可以清晰地看到喷气机产生的效果。

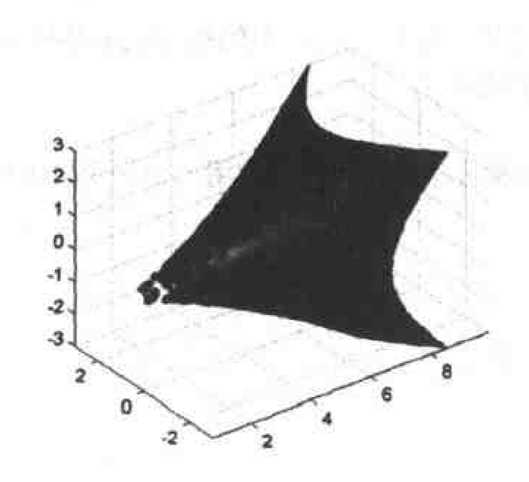


图6-37 不透明情况下水流速度的显示效果

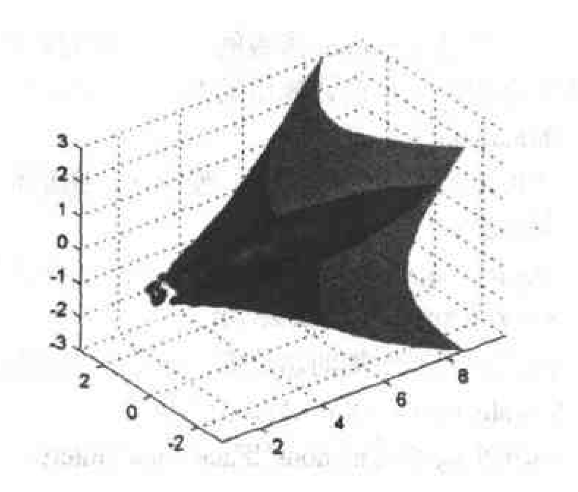


图6-38 增加透明度后水流的的显示效果

如果AlphaDataMapping属性设置为none，那么通过设置AlphaData可以使整个图形使用指定的α值着色。

6.3.5 透明度数据映射

α值数据类似于颜色数据：当用户创建一个曲面时，MATLAB将颜色数组的每一个元素映射为色彩映射表的一个颜色；同样，每一个α值数据也将被映射为α值映射表的一个透明度。

使用AlphaData属性指定曲面或图形的α值。对每一个面片，使用FaceVertexAlphaData属性设置小面或顶点数目。缺省情况下MATLAB不显示物体的透明度。由于α值不是一个标量，所以用户需要指定α值数组的大小使其与图形或曲面的颜色数组的大小相等。指定了α值数组的大小之后就可以选择用户希望使用的表面和边缘着色方式。Flat为每个表面使用统一的透明度，而interpolated为每一个顶点使用α值的双线形插值结果。

下面介绍如何将α值映射到α映射表。α映射表是一个数值从0到1的简单数组，该数组可以是m×1维的，也可以是1×m维的。缺省的α映射表包含64个从0到1等间距递增的数值。用户可以通过使用alphamap函数来调用预定义的映射表，当然用户也可以修改这些已存在的映射表，例如：

```
plot(alphamap('vup'))
```

可以看出vup是如图6-39所示的一个映射表。

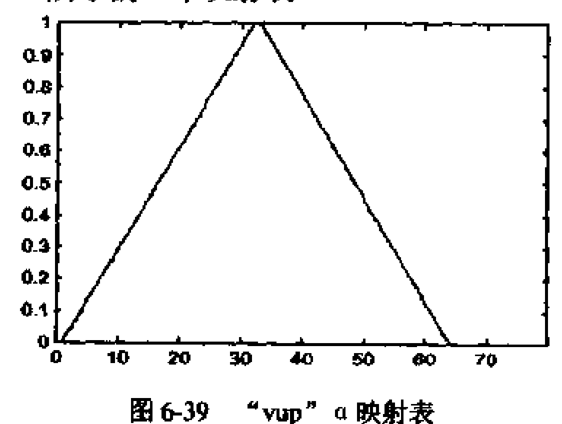


图 6-39 “vup” α映射表

可以使用alphamap函数的可选参数对映射表进行修改，使表中的每个元素都增大0.4，如果映射表中元素的计算结果大于1，则该元素取值为1：

```
alphamap('increase',.4)
```

下面给出一个映射实例：使用切片图形来检验音量数据。首先使用一个三变量函数来生成音量数据：

```
[x,y,z] = meshgrid(-1.25:.1:-.25,-2:.2:2,-2:.1:2);
v = x.*exp(-x.^2-y.^2-z.^2);
```

然后创建该数据的切片图，指定使用插值映射方式：

```
h = slice(x,y,z,v,[-1 -.75 -.5],[],[0]);
set(h,'EdgeColor','none','FaceColor','interp',...
'FaceAlpha','interp')
```

将 α 值设置为与颜色值相等的数值，调用一个递减的 α 映射表（最小的数值代表最大的透明度）并给该表的每一个元素增加0.1来实现所需的透明度。本例中使用的颜色映射表为hsv：

```
alpha('color')
alphamap('rampdown')
alphamap('increase',.1)
colormap(hsv)
```

绘图结果如图6-40所示，可以看到，音量数据分为三组保持在零点附近。

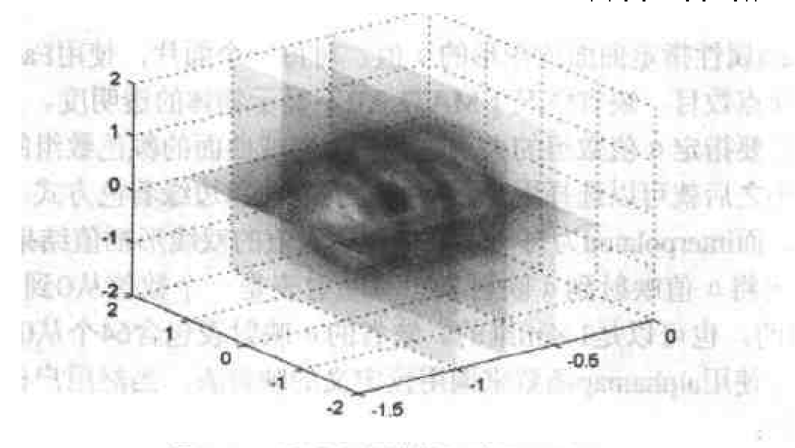


图 6-40　透明音频数据切片显示效果

6.4　实 例 讲 解

【实例】：绘制一个一阶六自由度球状谐波图形，给定幅值为 2，半径为 5。

分析：球状谐波是傅立叶级数的球状描述版本。通过使用球状谐波可以模拟地球的自由摆动行为。这里不介绍球状谐波的原理及生成函数，仅从图形绘制的角度来进行说明绘图的每一步骤。为了给图形添加真实感和立体感，在使用曲面绘图函数 surf 生成原始曲面的基础上，添加灯光并设置着色方式。最后通过不同角度来观察该图形。

解决方法：

步骤一：定义有关参数：

```
% 定义自由度
n = 6;
% 定义阶数
m = 1;
delta = pi/40;
```

步骤二：定义三维网格：

```
% 定义网格高度
theta = 0 : delta : pi;
% 定义方位角
phi = 0 : 2*delta : 2*pi;
[phi,theta] = meshgrid(phi,theta);
```

步骤三：曲面计算：

```
Ymn = legendre(n,cos(theta(:,1)));
Ymn = Ymn(m+1,:);
yy = Ymn;
for kk = 2: size(theta,1)
      yy = [yy Ymn];
end;
yy = yy.*cos(m*phi);
m = max(max(abs(yy)));
rho = 5 + 2*yy/m;
% 球坐标方程
r = rho.*sin(theta);
x = r.*cos(phi);
y = r.*sin(phi);
z = rho.*cos(theta);
```

步骤四：绘制图形：

```
surf(x,y,z)
```

步骤五：添加灯光并设置着色方式：

```
light
lighting phong
```

步骤六：设置坐标轴：

```
axis('square')
axis([-5 5 -5 5 -5 5])
axis('off')
```

绘图结果如图 6-41 所示。

步骤七：改变观察角度。改变视角：

```
view(40,30)
```

绘图结果如图 6-42 所示。



图 6-41　球状谐波



图 6-42　改变视角后的球状谐波

6.5 小　结

通过本章的学习，读者将充分领略到 MATLAB 图形系统的庞大功能，学会如何绘制立体感和真实感较强的三维曲面和场景。到目前为止，本书所介绍的绘图方法主要是利用 MATLAB 的高级绘图函数，在以后的学习中，读者将学到如何利用 MATLAB 的低级操作命令来定制所需的图形。

第 7 章　图形对象句柄

通过以上几个章的学习，读者已经掌握了如何使用 MATLAB 的高级绘图函数和工具来创建一幅二维或三维图形。在本章中，读者将学习 MATLAB 低级绘图命令的使用方法，我们首先对图形对象及其属性的概念进行介绍，然后说明如何利用图形对象创建函数来生成图形对象。通过学习图形对象句柄的使用方法，读者将会掌握如何通过设置对象的属性来获得所需的图形外观。

7.1　图形对象及对象属性

7.1.1　图形对象概述

图形对象句柄是 MATLAB 用来显示数据和创建图形用户接口的基本绘图元素。每一个对象实例都是由一个独一无二的标示符来表示的，这个标示符就称为句柄。通过使用句柄，用户可以对一个已存在的图形对象进行特征操作（即对图形属性进行操作）。用户也可以在创建图形时定义图形的属性。显然，要掌握句柄的使用方法首先要掌握图形对象的知识。

图形对象之间存在着一定的依赖关系，例如，假设用户希望绘制一个直线对象，那么 MATLAB 需要一个坐标轴对象对直线进行定向并为该直线提供一个参考框架，而坐标轴又相应地需要一个图形窗口来显示坐标轴轴线。根据图形对象间的这种依赖关系，MATLAB 将所有图形对象组织在一个树形结构的层次关系表（参见图 3-3）中。与其他图形对象关联越紧密的对象层次越高。

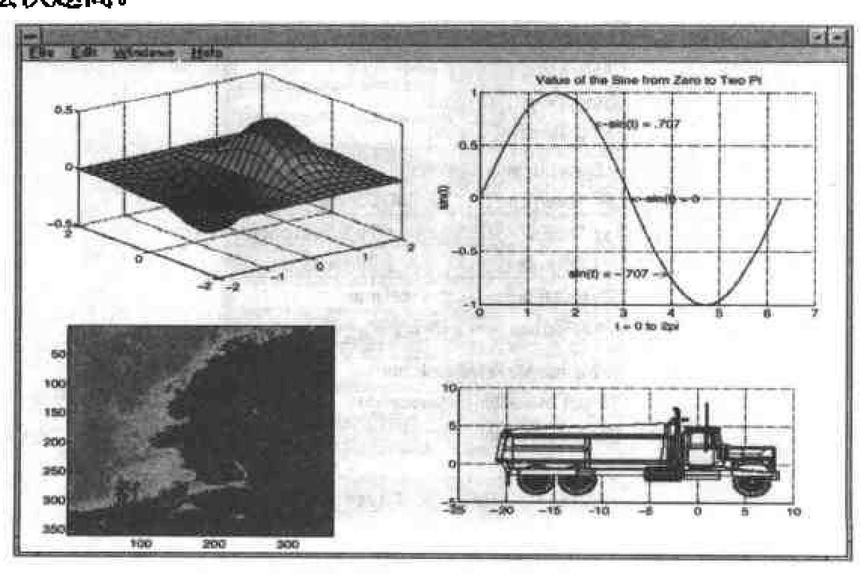


图 7-1　图形对象实例

图 7-1 给出了一个图形实例，该图形包括以下几种图形对象：图形窗口对象、曲面对象、二维坐标轴对象和三维坐标轴对象、线条对象、文本对象、图像对象、面片对象。

7.1.2　图形对象种类

根对象（root）：位于继承关系表的最顶端是 root 对象。root 相当于计算机的显示屏。一个 GUI 中仅有一个 root 对象，其他所有对象都是它的后裔。用户是不能创建 root 对象的，当用户启动 MATLAB 时 root 对象就已经存在。用户通过设置 root 的属性值改变图形的显示效果。

图形窗口对象（figure）：figure 对象是位于 root 显示屏上的单独窗口，MATLAB 在这个窗口中显示图形。MATLAB 对 figure 对象的数量没有限制，用户以创建任意多个图形窗口。所有的 figure 对象都是 root 的子对象，而除了 root 以外的所有其他对象都是 figure 的后裔。如果调用绘图函数时图形窗口不存在，所有的绘图函数都能够自动创建一个 figure 对象。如果在 root 中有许多个 figure 对象，其中有一个将被定义为"当前"窗口，作为图形输出的目标。

用户控件对象（uicontrol）：uicontrol 对象是一个用户接口控件，当用户激活这个对象时，该对象将执行相应的回调函数。uicontrol 控件包括按钮、列表框和滚动条等。每一个控件都被设计用来接收用户的某种特定信息。例如，列表框通常被用来提供一系列的文件名，用户可以从中选择一个或多个选项，整个选择过程由控件的回调程序来完成。用户可以联合使用多个 uicontrol 对象来构造 GUI 界面和对话框。图 7-2 所示的例子就是由弹出菜单、编辑框、复选框、按钮、文本和框架等多个用户控件组成的。图中弹出式菜单的作用是提供预先给定项目的选择功能，编辑框用于接收用户输入的数值，复选按钮允许用户选择多个特殊选项，复选框提供其他控件的逻辑组，静态文本作为用户控件的标签，按钮将确认用户的某种行为。所有的 uicontrol 对象都是 figure 对象的子对象。

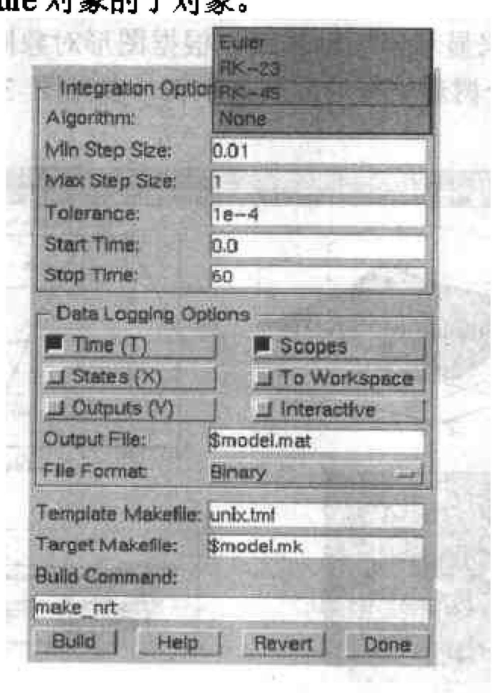


图 7-2　用户控件对象实例

用户菜单对象（uimenu）：uimenu 对象通过执行相应的回调函数来完成用户选择某个菜单选项希望进行的操作。MATLAB 将 uimenu 对象放在图形窗口的菜单栏上，按顺序排列在系统菜单的右边。图 7-3 中的 Windows 系统图形窗口包含三个 uimenu 对象（菜单）：Workspace，Figure，Axe。Workspace 有两级可见的子菜单。

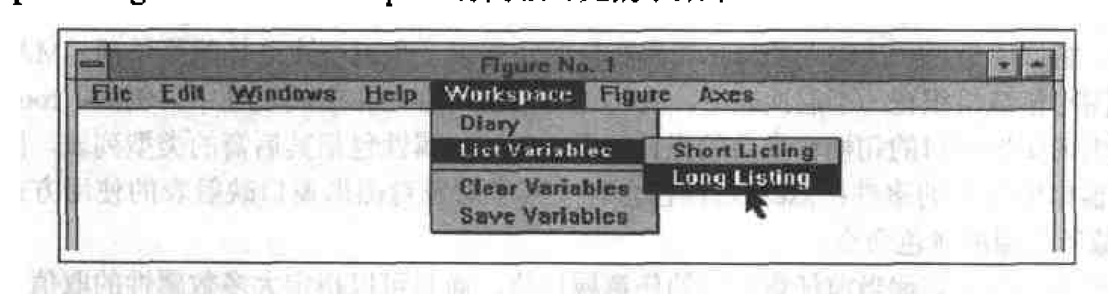


图 7-3　用户菜单对象实例

坐标轴对象（axes）：axes 对象在图形窗口中定义一个区域并使用这个区域为自己的子对象定向。axes 是 figure 的子对象，是 image、light、line、patch、surface 和 text 对象的父对象。如果创建图形时不存在 axes 对象，那么所有的绘图函数（例如 plot，surf，mesh 和 bar）将会自动创建一个 axes 对象；如果存在多个 axes 对象，则指定其中一个为“当前” axes 对象，以该对象为 axes 子对象的显示目标。

图像对象（image）：image 对象由一个数据矩阵和颜色映射（可不存在）组成。图像可以分为三种类型：真彩，索引和高度，不同图像类型间的区别在于数据矩阵与像素颜色的关系不同。由于图像是一个严格二维图形，因而用户仅可以在二维视图中观察图像。

灯光对象（light）：light 对象定义光源，该光源对坐标轴中所有的 patch（面片）和 surface（曲面）对象产生影响。虽然用户看不到光源，但是可以通过设置光源的属性（颜色、位置及其他）来控制光源。

线条对象（line）：line 对象是创建大多数二维和三维绘图的基本图形元素。高级函数 plot，plot3 和 loglog 等可以用来创建 line 对象。axes 对象的坐标系统决定 line 对象的位置和方向。

面片对象（patch）：patch 对象由有边缘的多边形构成。单个的 patch 对象可以有许多个小面，每个小面可以独立地使用均匀或插值着色方式。fill、fill3 和 contour3 函数可以用来创建 patch 对象。patch 同样由 axes 的坐标系统定位并定向。

矩形对象（rectangle）：rectangle 对象是一个二维填充区域，其形状可以是矩形或椭圆形。rectangle 对象一般用来绘制流程图。

曲面对象（surface）：surface 对象是矩阵数据的三维描述图形，使用矩阵元素作为点到 x-y 平面的高度，通过调用绘图函数绘制而成。曲面图形由多个四边形组成，四边形的顶点由矩阵数据决定。MATLAB 可以使用均匀或插值着色方法对曲面进行着色，也可以仅使用网格线将相关点连接起来而不对四边形表面着色。axes 对象的坐标系统决定曲面的位置和方向。函数 pcolor 和 surface、mesh 函数可以用来创建 surface 对象。

文本对象（text）：text 对象即字符串。由 axes 对象的坐标系统定义字符串的位置。高级函数 title、xlabel、ylabel、zlabel 和 gtext 将创建 text 对象。

7.1.3　图形对象属性概念

图形对象的属性可以控制对象外观和行为的许多特征。属性不但包括对象的一般信息，而且包括特殊类型对象独一无二的信息。例如，用户可以从任意给出的 figure 对象中获得以下信息：窗口中最后一次输入的标示符、指针的位置以及最近一次选择的菜单项。MATLAB 将所有图形信息组织在一个层次表中，并将这些信息储存在相应的属性中。例如，root 属性表包括当前图形窗口的句柄和当前的指针位置；figure 属性包括其后裔的类型列表，同时实时跟踪窗口中发生的事件；axes 属性包含有关其子对象对图形窗口映射表的使用方式以及 plot 函数所使用的颜色命令。

用户不但可以查询当前任意对象的任意属性值，而且可以指定大多数属性的取值（某些属性为 MATLAB 控制的只读属性）。属性值仅对对象的特定实例起作用，也就是说，修改属性值不会对同类对象、不同实例的属性值产生影响。可以通过设置属性的缺省值来影响所有此后创建对象的属性。如果用户既没有定义缺省值又没有在创建对象时指定属性值，MATLAB 将使用“出厂”时定义的属性值，该属性称为系统缺省值。每个对象创建函数的参考入口都提供一个与图形对象有关的属性值完整列表。

有些属性是所有图形对象都具备的，例如类型（Type）、被选状态(Selected)、是否可见(Visible)和创建回调函数 CreateFcn、销毁回调函数 DeleteFcn（参见表 3-3）。而有些属性则是某种对象独有的，例如线条对象的线型属性等。这些独有的属性将在介绍属性设置方法时予以介绍。

在不引起混淆的前提下，编程时允许使用属性名的缩写。但是，在编写 M 文件时最好不使用缩写，以防将来MATLAB 系统扩展时导致属性重名现象。

set 和 get 函数可以指定或获取已存在对象的属性值，如果该属性值有一个取值范围集，这两个函数还能够将该属性所有可能的取值列举出来。设置已存在对象属性的基本语法格式如下（属性名由单引号括起）：

```
set(object_handle,'PropertyName','NewPropertyValue')
```

如果希望查询指定对象的当前属性值，可以使用以下语句：

```
returned_value = get(object_handle,'PropertyName');
```

7.2　图形对象操作方法

7.2.1　创建图形对象

典型的图形通常包括许多种相关的图形对象，由这些对象共同生成有具体含义的图形或图片。每一种类型的图形对象都有一个相应的创建函数（除了 root 对象），这个创建函数使用户能够创建该对象的一个实例。对象创建函数名与所创建的对象名相同，例如，函数 text 将创建一个 text 对象，figure 函数将创建一个 figure 对象。表 7-1 列出了 MATLAB 所有的对象创建函数。

表 7-1 图形对象创建函数及其创建对象描述

函 数	对 象 描 述
Axes	标度和定向 axes 子对象 image、light 等的矩形坐标系统
figure	显示图形的窗口
image	使用颜色映射表索引或 RGB 值的二维图片。数据可以是 8 位也可以是双精度数据。
Light	位于坐标轴中，能够影响面片和曲面的有方向光源
Line	由顺序连接坐标数据的直线段构成的线条
patch	将矩阵的每一列理解为由一个多边形构成的小面
rectangle	矩形或椭圆形的二维填充区域
surface	由矩阵数据（理解为高度）定义的矩形面创建而成的曲面
Text	位于坐标轴系统中的字符串
uicontextmenu	与其他图形对象相关的用户文本菜单
uicontrol	可编程用户接口控件，例如按钮、滚动条和列表框等
uimenu	在图形窗口顶端出现的菜单

所有的对象创建函数都有相同的调用格式：

返回句柄=创建函数名（'属性名'，属性值，……）

用户可以使用属性名/属性值参数对给任意的对象属性指定一个数值（除了只读类型属性以外）。函数将返回创建对象的句柄，在这之后用户可以使用这个句柄查询或修改所创建对象的属性值。下面给出一个创建对象的实例：首先对一个数学函数求值，再使用 figure、axes 和 surface 函数创建三个图形对象并设置其属性，其他图形对象的属性使用 MATLAB 缺省值。

```
[x,y] = meshgrid([-2:.4:2]);
Z = x.*exp(-x.^2-y.^2);
fh = figure('Position',[350 275 400 300],'Color','w');
ah = axes('Color',[.8 .8 .8],'XTick',[-2 -1 0 1 2],...
        'YTick',[-2 -1 0 1 2]);
sh = surface('XData',x,'YData',y,'ZData',Z,...
        'FaceColor',get(ah,'Color')+.1,...
        'EdgeColor','k','Marker','o',...
        'MarkerFaceColor',[.5 1 .85]);
```

注意 surface 函数并不是像 surf 函数那样使用三维的视图。绘图结果如图 7-4 所示。

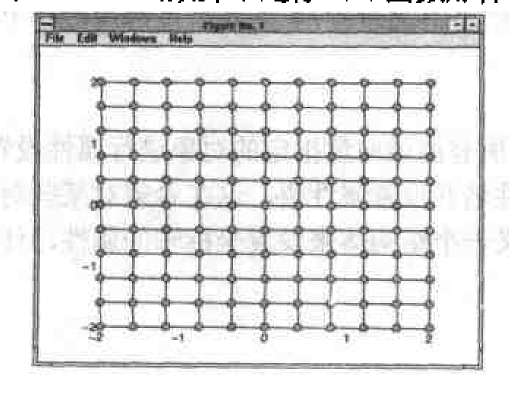


图 7-4 创建图形对象实例

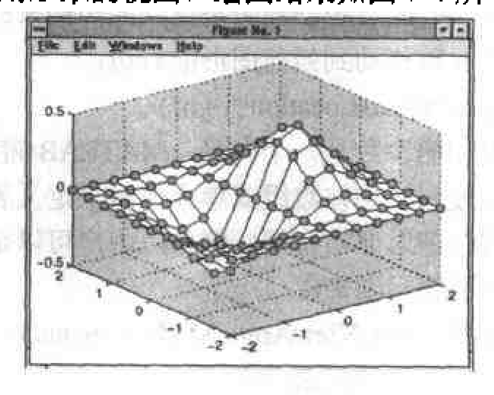


图 7-5 视图改变后的图形效果

为了清楚地观察创建的各个图形对象，可以通过镜头命令或 view 命令改变视角：

```
view(3)
```

改变视角后的图形外观如图 7-5 所示。

对象创建函数还能够使用一种较为简单的调用形式，例如

```
text(.5,.5,.5,'Hello')
```

与：

```
text('Position',[.5 .5 .5],'String','Hello')
```

是等效的。注意，使用对象创建函数的这种形式有时会导致输出结果与正常调用形式的结果有细微的不同。

缺省情况下，所有的创建函数都将当前的图形窗口和坐标轴作为其创建对象的父对象。用户也可以指定所创建对象的父对象。例如：

```
axes('Parent',figure_handle,...)
```

将在 figure_handle 指定的图形窗口中创建一个坐标轴。用户也可以通过定义 parent 属性将所创建的对象从一个父对象中转移到另一个父对象中：

```
set(gca,'Parent',figure_handle)
```

实际上，MATLAB 高级绘图函数（例如 plot、surf 等）也是通过调用相应的创建函数来绘制图形对象的。高级函数可能会根据坐标轴和图形窗口的 NextPlot 属性来判断是否清除坐标轴或创建新的图形窗口；而低级对象创建函数则不考虑图形窗口和坐标轴的 NextPlot 属性，只是创建各自的图形对象并将这些对象放置在当前的父窗口中。例如，第一次调用：

```
line('XData',x,'YData',y,'ZData',z,'Color','r')
```

将使用指定的数值在当前坐标轴中绘制一条红线（如果不存在坐标轴，MATLAB 将创建一个坐标轴；如果不存在图形窗口，MATLAB 也将创建一个图形窗口）。如果用户第二次调用 line 函数，MATLAB 将在当前坐标轴中绘制第二条直线，但并不擦除第一条直线。高级绘图函数（例如 plot）则与之不同，它们将擦除以前的图形对象并重置所有的坐标轴（除了位置和比例尺）。用户通过使用 hole 命令或通过改变 NextPlot 属性来改变高级绘图函数的这种行为。

7.2.2　图形对象属性设置

在创建图形对象的同时，用户可以根据自己的需要来设置相应图形对象的属性。用户还可以通过 set 函数和创建函数返回的句柄来改变一个已存在对象的属性。例如，以下语句将坐标轴的 y 轴移动到当前图形的右边：

```
set(gca,'YAxisLocation','right')
```

如果句柄参数是一个向量，MATLAB 将对所有由该向量指定的对象进行属性设置。

可以使用结构体数组或单元数组来定义属性名并设置属性值，这在希望对某些对象使用相同的属性值时非常有用。例如，用户可以定义一个结构体来设置坐标轴的属性，使之符合特定图形的显示要求：

```
view1.CameraViewAngleMode = 'manual';
view1.DataAspectRatio = [1 1 1];
view1.ProjectionType = 'Perspective';
```

为了设置当前坐标轴的这些属性值，输入语句：

```
set(gca,view1)
```

还可以使用 set 函数来显示多个属性的可能取值但不真正设置新的属性值。例如，以下语句将获得 line 对象属性的可能取值：

```
set(obj_handle,'Marker')
```

MATLAB 将返回由 obj_handle 定义的 Marker 属性的数值列表：

```
[ + | o | * | . | x | square | diamond | v | ^ | > | < | pentagram| hexagram | {none} ]
```

如果希望得到所有可设置的属性，或者希望得到能够接收字符串的属性的所有可能取值，使用对象句柄作为 set 函数的惟一参数即可实现：

```
set(object_handle)
```

例如，对于一个曲面对象调用以上语句，MATLAB 将返回如下信息：

```
CData
CDataScaling: [ {on} | off]
EdgeColor: [ none | {flat} | interp ] ColorSpec.
EraseMode: [ {normal} | background | xor | none ]
FaceColor: [ none | {flat} | interp | texturemap ] ColorSpec.
LineStyle:[{-}|--|:|-. none ]
.Visible: [ {on} | off ]
```

set 函数的输出是一个结构体数组。例如：

```
a = set(gca);
```

则返回值 a 是一个结构体，其域名是对象的属性名，域值是属性的可能取值。例如假设需要获得坐标轴网线格式的可能取值，使用以下语句：

```
a.GridLineStyle
```

MATLAB 将输出：

```
ans =
    '-'
    '--'
    ':'
    '-.'
    'none'
```

注意虽然属性名与大小写无关，但结构体域名与大小写有关，如果输入：

```
a.gridlinestyle
```

MATLAB 将产生一个错误信息。

7.2.3 属性值查询

使用 get 函数来查询一个或多个对象属性的当前取值。例如，使用以下语句检查当前坐标轴 PlotBoxAspectRatio 属性的取值：

```
get(gca,'PlotBoxAspectRatio')
ans =
```

1 1 1

虽然 MATLAB 能够列举所有属性的取值，但对于包含数据的属性，例如 Current 和 ColorOrder，MATLAB 仅列举该属性值的维数。例如：

```
AmbientLightColor = [1 1 1]
Box = off
CameraPosition = [0.5 0.5 2.23205]
CameraPositionMode = auto
CameraTarget = [0.5 0.5 0.5]
CameraTargetMode = auto
CameraUpVector = [0 1 0]
CameraUpVectorMode = auto
CameraViewAngle = [32.2042]
CameraViewAngleMode = auto
CLim: [0 1]
CLimMode: auto
Color: [0 0 0]
CurrentPoint: [ 2x3 double]
ColorOrder: [ 7x3 double]
Visible = on
```

用户可以使用指定的属性名单独查询某一属性，从而获得该属性的具体数据：

```
get(gca,'ColorOrder')
ans =
    0 0 1.0000
    0 0.5000 0
    1.0000 0 0
    0 0.7500 0.7500
    0.7500 0 0.7500
    0.7500 0.7500 0
    0.2500 0.2500 0.2500
```

如果用户将输出为指定某一变量，则 MATLAB 将该变量作为一个结构体数组进行赋值，该数组的域名是对象的属性名，域值是属性名对应的当前取值。例如，如果用户希望绘制数据 x 和 y：

```
h = plot(x,y);
```

并获得由 plot 函数创建的 line 对象的所有属性：

```
a = get(h);
```

下面用户就可以通过使用 a 的域名来访问 line 属性的数值。以下语句调用 text 函数将字符串 “x and y” 作为数据 x 和 y 的起点，文本使用与 line 相匹配的颜色：

```
text(x(1),y(1),'x and y data','Color',a.Color)。
```

如果 x 和 y 是矩阵，plot 函数将为矩阵的每一列绘制一条线。为了标记数据第二列产生的线条，使用以下语句：

text(x(1,2),y(1,2),'Second set of data','Color',a(2).Color)。

如果希望查询属性群，可以定义一个属性名单元数组并使用它方便地获取这些属性的取值。例如，假设用户希望查询坐标轴 camera mode 的属性，定义以下的单元数组：

```
camera_props(1) = {'CameraPositionMode'};
camera_props(2) = {'CameraTargetMode'};
camera_props(3) = {'CameraUpVectorMode'};
camera_props(4) = {'CameraViewAngleMode'};
```

使用这个单元数组作为参数来获得这些属性的当前取值：

```
get(gca,camera_props)
ans =
'auto' 'auto' 'auto' 'auto'
```

7.2.4　设置用户属性缺省值

如果用户没有指定属性值，同时又禁止使用缺省的属性值，那么 MATLAB 将使用系统默认属性值为所有对象定义属性值。用户可以使用以下语句获得所有属性的系统默认属性值：

```
a = get(0,'Factory');
```

get 函数将返回一个结构体数组，该数组的域名由对象类型和属性名联合构成，域值是域名所表示的对象和属性的系统默认属性值。例如：

UimenuSelectionHighlight: 'on'

表示 uimenu 对象 SelectionHighlight 属性的系统默认属性值为 on。

可以使用以下语句获得单个属性的系统默认属性值：

```
get(0,'FactoryObjectTypePropertyName')
```

例如：

```
get(0,'FactoryTextFontName')
```

所有对象的属性都有一个 MATLAB 系统默认属性值。

在进行绘图或其他需要了解对象属性值的工作时，MATLAB 将从当前对象开始，在继承表中始终向上搜索，直到找到用户定义的缺省值或系统默认属性值，因此总能找到合适的属性值。可以看出，用户定义的缺省值越靠近继承表的 root 对象，MATLAB 的搜索范围越广。如果用户在 root 级定义 line 对象的缺省值，MATLAB 将对所有的 line 对象使用这个缺省值，如果用户仅在 axes 级定义 line 对象的缺省值，则 MATLAB 将对该坐标轴内的所有 line 对象使用该缺省值。如果用户在多级中定义对象的缺省值，与该对象最近的父对象级中定义的缺省值将被使用，因为该缺省值是 MATLAB 最先找到的缺省值。

注意缺省值的设置仅对那些设置完成后所创建的对象有效，已存在的图形对象不会发生变化。指定缺省值首先要创建一个以 Default 开头、然后紧跟对象类型、最后是对象属性的字符串。例如，如果希望在当前的图形窗口级指定 line 对象的 LineWidth（线性的宽度）属性为 1.5 个点宽，使用以下语句：

```
set(gcf,'DefaultLineLineWidth',1.5)
```

字符串 DefaultLineLineWidth 将指明该属性是 line 对象的 LineWidth 属性。定义图形窗

口的颜色可以使用字符串 DefaultFigureColor。这里要注意的是，仅在 root 级指定图形窗口的属性才是有意义的。

```
set(0,'DefaultFigureColor','b')
```

可以使用 get 函数来查询当前所有级的缺省值设置情况：

```
get(gcf,'default')
```

使用属性值“default”可以将一个属性设置为 MATLAB 第一个搜索到的缺省值。例如，以下语句将产生一个绿色的曲面 EdgeColor 属性（边缘颜色）：

```
set(0,'DefaultSurfaceEdgeColor','k')
h = surface(peaks);
set(gcf,'DefaultSurfaceEdgeColor','g')
set(h,'EdgeColor','default')
```

在 figure 级存在一个该属性的缺省值，MATLAB 将第一个找到这个缺省值，于是用它来替换 root 级定义的缺省值。

指定某个属性的数值为“remove”将会删除用户定义的缺省值：

```
set(0,'DefaultSurfaceEdgeColor','remove')
```

以上语句将 root 级定义的曲面 EdgeColor 缺省值删除。

指定一个属性取值为“factory”，则该属性采用系统默认属性值。例如，以下语句将曲面的 EdgeColor 属性设置为系统默认属性值颜色——黑色，而用户设置的缺省值将被忽视：

```
set(gcf,'DefaultSurfaceEdgeColor','g')
h = surface(peaks);
set(h,'EdgeColor','factory')
```

注意在使用对象创建函数时，default、remove 和 factory 必须以反斜线开头说明该字符串为保留字，例如：

```
h = uicontrol('Style','edit','String','\Default');
```

下面给出两个例子来说明以上内容。plot 函数在显示多个图形时将循环使用由坐标轴的 ColorOrder 属性定义的颜色。如果用户为坐标轴的 LineStyleOrder 属性定义多个属性值，那么 MATLAB 将在每一次颜色循环后改变线型的宽度。用户还可以通过定义属性缺省值使 plot 函数使用不同的线型来创建图形。下面给出几个通过修改属性值进行绘图的例子。

例一：使用白底色创建一个图形窗口，然后在 root 级为坐标轴对象设置缺省值：

```
whitebg('w') %create a figure with a white color scheme
set(0,'DefaultAxesColorOrde r', [0 0 0],...
'DefaultAxesLineStyleOrder','-|--|:|-.'
```

无论何时用户调用 plot 函数，

```
Z = peaks; plot(1:49,Z(4:7,:))
```

该函数将采用同一种颜色绘制所有数据，这是因为坐标轴的 ColorOrder 属性仅包括一个颜色。但是函数将循环使用 LineStyleOrder 定义的不同线型。

例二：在继承表的多级上设置缺省值。以下语句在一个图形窗口中创建两个坐标轴，分别在 figure 级和 axes 级设置坐标轴颜色、线型等属性的缺省值。

```
t = 0:pi/20:2*pi;
s = sin(t);
```

```
c = cos(t);
% 设置坐标轴颜色属性缺省值
figh = figure('Position',[30 100 800 350],...
     'DefaultAxesColor',[.8 .8 .8]);
axh1 = subplot(1,2,1); grid on
% 在第一个坐标轴处设置线型缺省值
set(axh1,'DefaultLineLineStyle','-.')
line('XData',t,'YData',s)
line('XData',t,'YData',c)
text('Position',[3 .4],'String','Sine')
text('Position',[2 -.3],'String','Cosine',...
     'HorizontalAlignment','right')
axh2 = subplot(1,2,2); grid on
% 在第二个坐标轴处设置文本旋转属性的缺省值
set(axh2,'DefaultTextRotation',90)
line('XData',t,'YData',s)
line('XData',t,'YData',c)
text('Position',[3 .4],'String','Sine')
text('Position',[2 -.3],'String','Cosine',...
     'HorizontalAlignment','right')
```

此时，不同的子图形使用相同的 text 和 line 函数会得到不同的显示结果(如图 7-6 所示)。

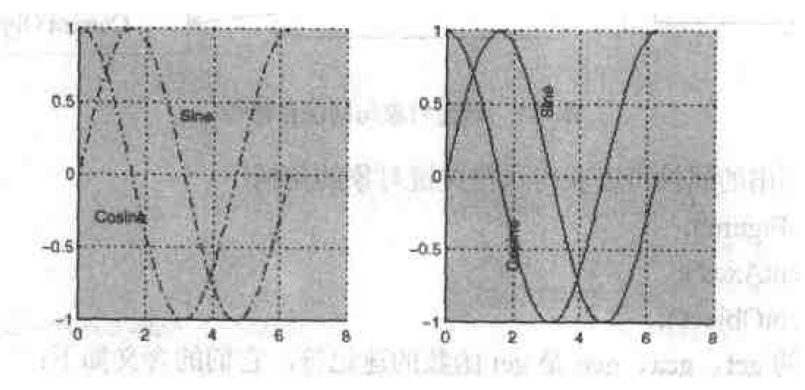


图 7-6　使用不同缺省属性值的不同效果

由于缺省的坐标轴颜色（Color）属性是在图形窗口级（figure）设置的，所以 MATLAB 使用相同的灰色背景颜色创建坐标轴。左图的坐标轴将点划线线型作为缺省线型，所以该图中每一次调用 line 函数都使用这种线型，而右边的坐标轴没有定义缺省线型，所以 MATLAB 使用系统默认属性值——实线型。右边的坐标轴给文本的 Rotation 属性定义为 90o，该图中所有的文本都使用这种旋转角度，而左边的坐标轴使用固有缺省值，不旋转文本。

如果用户希望无论何时运行 MATLAB 都使用用户定义的缺省值，可以在 startup.m 文件中指定这些缺省值。注意，在调用 colordef 命令时，MATLAB 可能会自动装载一些外观属性的系统缺省值。

7.3　句柄使用方法

7.3.1　访问对象句柄

MATLAB 给所创建的每一个图形对象指定一个句柄。所有的对象创建函数都能够返回被创建对象的句柄。如果用户希望访问对象的属性，那么最好在创建对象时将对象的句柄赋给一个变量，以免以后对句柄进行重复搜索。当然，用户也可以使用 findobj 函数或通过查询其父对象的 Children 属性来获得已存在对象的句柄。

除了 root 对象和 figure 对象以外，所有对象的句柄都是一个浮点数。用户使用这些句柄数据时必须保证这些数据的全部精度。root 对象的句柄总是 0；figure 对象的句柄可以是显示在窗口标题栏中的整数（缺省情况下），也可以是一个完全符合 MATLAB 内部浮点数精度的数值。

图形对象的所有操作都是针对当前对象而言的。“当前”是图形对象的一个重要概念，当前窗口是指被指定作为图形输出的窗口；当前坐标轴是创建坐标轴子对象命令的目的坐标轴；当前对象是用户创建的最后一个图形对象或鼠标点击的最后一个对象。

MATLAB 分别将这些当前对象的句柄保存在其父对象的属性中（参见图 7-7）。

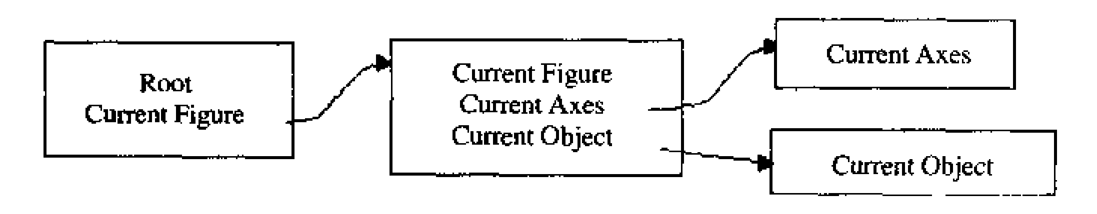


图 7-7　当前对象句柄保存路径

从图 7-7 中列出的属性可以获得这些关键对象的句柄：

```
get(0,'CurrentFigure');
get(gcf,'CurrentAxes');
get(gcf,'CurrentObject');
```

以上语句中的 gcf、gca、gco 是 get 函数的速记符，它们的含义如下：

gcf：返回 root 的 CurrentFigure 属性值；

gca：返回当前图形窗口的 CurrentAxes 属性值；

gco：返回当前图形窗口的 CurrentObject 属性值。

用户可以使用这些速记符（也可以称为命令）作为函数的对象句柄参数。例如，用户可以点击一个 line 对象，然后使用 gco 作为 set 函数的参数来指定该 line 对象的句柄：

```
set(gco,'Marker','square')
```

或者使用以下语句列举所有当前坐标轴的属性值：

```
get(gca)
```

使用以下语句可以获得当前坐标轴中的所有图形对象的句柄：

```
h = get(g ca,'Children');
```

通过以下语句确定对象的类型：

```
get(h,'type')
ans =
    'text'
    'patch'
    'surface'
    'line'
```

虽然 gcf 和 gca 提供了一个简单地获取当前图形窗口和坐标轴句柄的方法，但是很少在 M 文件中使用这两个命令，因为一般设计 MATLAB 程序 M 文件时不会根据用户行为来获得当前对象。

MATLAB 还提供一种通过属性值搜索对象的方法——findobj 函数。findobj 函数能够快速地形成一个继承表的横截面并获得具有指定属性值的对象句柄。如果用户没有指定一个开始搜索的对象，findobj 函数将从 root 对象开始，始终搜索与用户指定属性名和属性值相符的所有事件。举例说明该函数的使用方法。

图 7-8 所示的 sin 函数图形中包括用来标注特殊点的文本对象。

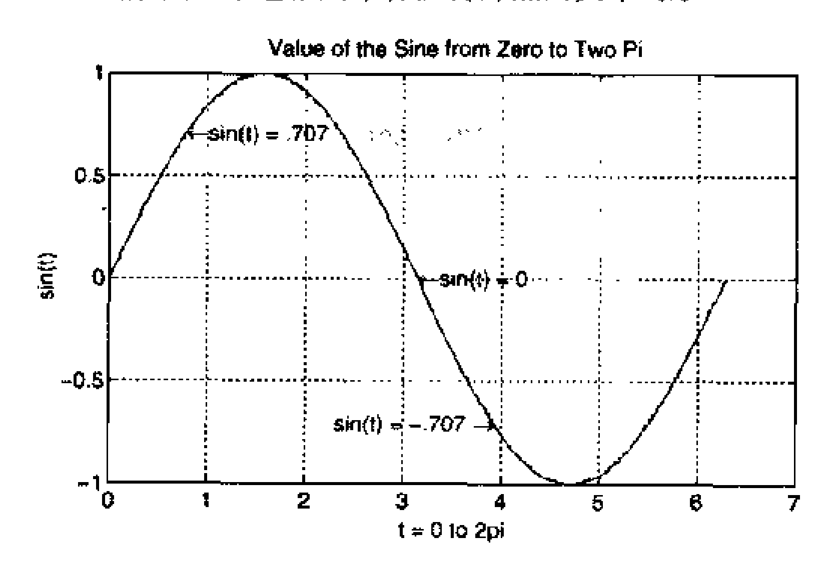


图 7-8　含有文本对象的图形

假设用户希望将文本字符串 sin(t)=.707 从当前位置移动到[3*pi/4, sin(3*pi/4)]。首先要获得文本对象的句柄，然后通过使用这个句柄修改文本对象的属性 Position。

为了使用函数 findobj，选择一个特殊的、能够惟一标示该文本对象的属性值来搜索当前对象的句柄。在本例中使用文本对象的 String 属性值。

```
text_handle = findobj('String','\leftarrowsin(t) = .707');
```

然后重新定义文本的 Position 属性值，将该对象移动到新的位置：

```
set(text_handle,'Position',[3*pi/4,sin(3*pi/4),0])
```

findobj 函数也允许用户通过指定搜索起始点来限制搜索，当继承表中有许多对象时，这样做将会大大提高搜索的速度。在本例中，假设用户知道所需的文本对象位于当前的坐标轴中，可以用以下语句进行快速搜索：

```
text_handle = findobj(gca,'String','\leftarrowsin(t) = .707');
```

7.3.2　使用句柄操作图形对象

1. 对象拷贝

用户可以使用 copyobj 函数将一个对象从一个父对象拷贝到另一个父对象中。新对象与旧对象惟一的不同就是其 Parent 属性和句柄。可以同时将多个对象拷贝到一个新的父对象中，也可以将一个对象拷贝到多个父对象中而不改变当前的父子关系。当用户拷贝一个有子对象的对象时，MATLAB 同时也拷贝其所有的子对象。

假设用户正在绘制多个数据并且希望标注每个图形点(5π/4,sin(5π/4))。text 函数将使用字符串{5π÷4,sin(5π÷4)}和一个右箭头来标注数据点，并将 HorizontalAlignment 属性修改为 right 使得数据点位于文本字符串的右边。

```
text('String','\{5\pi\div4, sin(5\pi\div4)\}\rightarrow',...
        'Position',[5*pi/4,sin(5*pi/4),0],...
        'HorizontalAlignment','right')
```

为了使用同样的字符串对另外一个点进行标注，首先使用 copyobj 函数拷贝文本对象。由于以上的语句并没有保存文本对象的句柄，所以用户必须使用 findobj 函数和 String 属性获得句柄。

```
text_handle = findobj('String',...
        '\{5\pi\div4,sin(5\pi\div4)\}\rightarrow');
```

在创建第二个图形后，通过从第一个图形中拷贝标签来为第二个图形添加标签：

```
copyobj(text_handle,gca).
```

2. 对象删除

用户可以使用 delete 命令和句柄参数删除一个图形对象。例如，用户可以使用以下语句删除当前的坐标轴（以及其所有子对象）：

```
delete(gca)
```

也可以使用 findobj 来获得需要删除的特定对象的句柄。

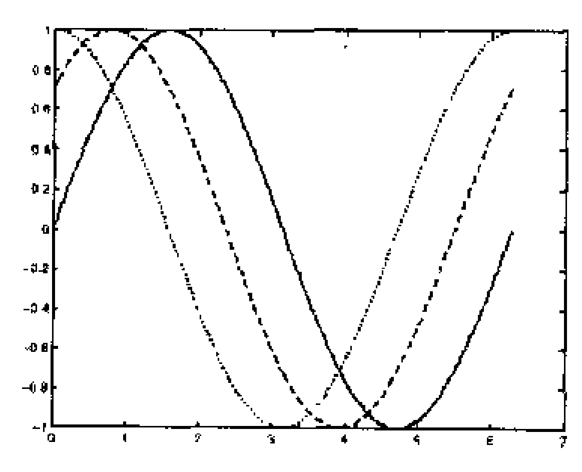


图 7-9　待删除的点线图形

例如，假设希望删除图 7-9 中的点线，首先获取点线的句柄：

```
line_handle = findobj('LineStyle',':');
```

然后使用这个句柄删除该线：

delete(line_handle)

其实以上两个语句可以合并为一个：

delete(findobj('LineStyle',':'))

删除结果如图 7-10 所示。

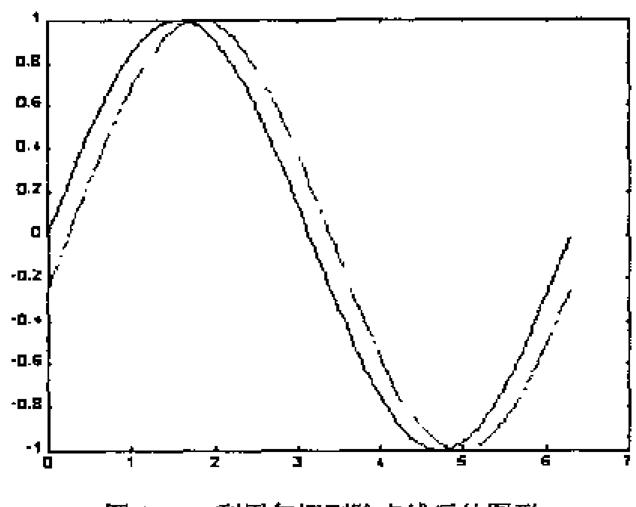


图 7-10　利用句柄删除点线后的图形

7.3.3　控制图形输出

MATLAB 允许同一次运行过程打开多个图形窗口，所以当一个 MATLAB 程序将创建图形窗口来显示图形用户界面并绘制数据时，有必要对某些图形窗口进行保护以免成为图形输出的目标，而相应的输出窗口要做好接受新图形的准备。以下内容将讨论如何使用句柄来控制 MATLAB 显示图形的目标和方式。

1. 指定输出目标

缺省情况下，MATLAB 图形创建函数在当前的图形窗口和坐标轴中显示图形。用户可通过在图形创建函数中使用明确的 Parent 属性直接指定图形的输出位置。例如：

plot(1:10,'Parent',axes_handle)

axes_handle 是目的坐标轴的句柄。

uicontrol 和 uimenu 函数指定 Parents 属性的语法比较简单：

uicontrol(Figure_handle,...)

uimenu(parent_menu_handle,...)

缺省情况下，产生图形输出的函数将在当前的图形窗口中显示该图形而不擦除或重置当前窗口的属性。然而，如果图形对象是坐标轴的子对象，MATLAB 为了显示这些图形将会擦除坐标轴并重置坐标轴的大多数属性。用户可以通过设置图形窗口和坐标轴的 NextPlot 属性来改变 MATLAB 的这种行为。MATLAB 高级图形函数在绘制图形前先要检查 NextPlot 属性，然后来决定是添加还是擦除重置图形和坐标轴。而低级对象创建函数则不检查 NextPlot 属性，只是简单地在当前窗口和坐标轴中添加新的图形对象。

表 7-2 列出了 NextPlot 属性的可能取值。

表 7-2　NextPlot 属性的可能取值

NextPlot	图 形 窗 口	坐 标 轴
Add	添加新的图形而不擦除或重置当前窗口	添加新的图形而不擦除或重置当前坐标轴
replacechildren	删除所有子对象但不重置窗口属性，等同于 clf 函数	删除所有子对象但不重置坐标轴属性，等同于 cla 函数
Replace	删除所有子对象并将窗口重置为缺省属性，等同于 clf 函数	删除所有子对象并将坐标轴重置为缺省属性，等同于 cla 函数

hold 命令提供访问 NextPlot 属性的方便方法，以下语句将图形和坐标轴的 NextPlot 属性都设置为 add：

hold on

以下语句将图形和坐标轴的 NextPlot 属性都设置为 replace：

hold off

MATLAB 提供 newplot 函数来简化图形 M 文件设置 NextPlot 属性的编写过程。newplot 函数首先检查 NextPlot 属性值，然后根据属性值决定相应的行为。用户应该在所有调用图形创建函数的 M 文件的开头定义 newplot 函数。当用户调用 newplot 函数时，有可能发生以下行为：

- 检查当前图形窗口的 NextPlot 属性，如果不存在图形窗口，则创建一个窗口并设该窗口为当前窗口；如果 NextPlot 值为 add，将该窗口设置为当前窗口；如果 NextPlot 值为 replacechildren，删除窗口的子对象并设置该窗口为当前窗口；如果 NextPlot 值为 replace，删除窗口的子对象，重置窗口属性为缺省值，并设置该窗口为当前窗口；
- 检查当前坐标轴的 NextPlot 属性：如果坐标轴不存在，创建一个坐标轴并将该坐标轴设置为当前坐标轴；如果 NextPlot 值为 add，将该坐标轴设置为当前坐标轴；如果 NextPlot 值为 replacechildren，删除坐标轴的子对象并设置该坐标轴为当前坐标轴；如果 NextPlot 值为 replace，删除坐标轴的子对象，重置坐标轴属性为缺省值，并设置该坐标轴为当前坐标轴。

缺省情况下，图形窗口的 NextPlot 值为 add，坐标轴的 NextPlot 值为 replace。

为了说明 newplot 的使用方法，下面给出一个类似于 plot 的绘图函数 my_plot，该函数在绘制多个图形时将循环使用不同的线型，而不是使用不同的颜色。

```
function my_plot(x,y)
  % newplot返回当前坐标轴的句柄
  cax = newplot;
  LSO = ['- ';'--';': ';'-.'];
  set(cax,'FontName','Times','FontAngle','italic')
  set(get(cax,'Parent'),'MenuBar','none') %
  line_handles = line(x,y,'Color','b');
  style = 1;
  for i = 1:length(line_handles)
  if style > length(LSO),
       style = 1;
```

```
end
   set(line_handles(i),'LineStyle',LSO(style,:))
   style = style + 1;
end
grid on
```

函数 my_plot 使用低级函数 line 语法来绘制数据，虽然 line 函数并不检查图形窗口和坐标轴的 NextPlot 属性值，但是由于 newplot 的调用使得函数 my_plot 与高级函数 plot 执行相同的操作：每一次用户调用该函数时，函数都对坐标轴进行清除和重置。my_plot 使用 newplot 函数返回的句柄来访问目标图形窗口和坐标轴。该函数还设置了坐标轴的字体属性并禁止使用图形窗口的菜单。调用 my_plot 函数的绘图结果如图 7-11 所示：

```
my_plot(1:10,peaks(10))
```

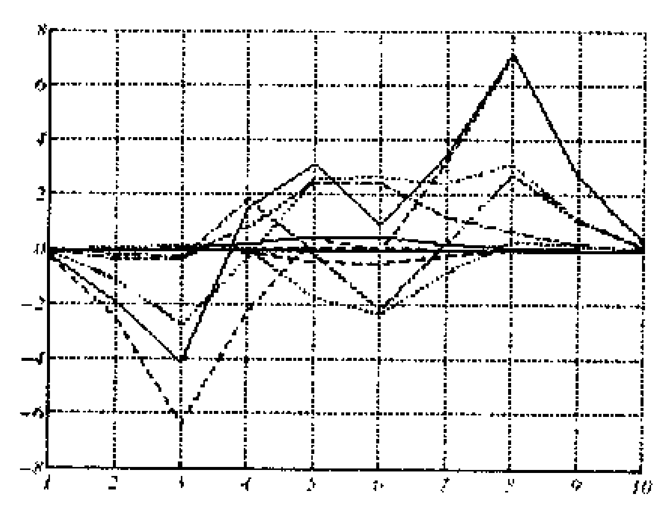


图 7－11　my_plot 函数的典型输出结果

2. 保护图形和坐标轴

有些情况下需要对图形窗口和坐标轴进行保护以免其成为图形输出的目标。用户可以将特定窗口或坐标轴的句柄从句柄列表中删除，使 newplot 和其他返回或参考句柄的函数（gca，gcf，gco，cla，clf，close 和 findobj）无法找到该句柄，从而保证该窗口不会成为其他程序的输出目标。有两个属性可以用来控制保护对象句柄的可见与否：HandleVisibility 和 ShowHiddenHandles。

HandleVisibility 是一个所有对象都具备的属性，该属性值可以为以下取值：

- on：对象句柄可以被任意函数获得。这是该属性的缺省值；
- callback：对象句柄对所有在命令行中执行的函数隐藏，而对所有回调函数（用户行为处理函数）总是可见的。这种可见程度将保证用户在命令行中键入的命令不会影响被保护对象；
- off：句柄对所有函数，无论是命令行中执行的还是回调函数，都是隐藏的。

例如，如果一个用户图形界面以文本字符串形式接受用户的输入，并在回调函数中对这个字符串进行处理，那么如果不对窗口进行保护，字符串“close all”有可能会导致用户图形界面的销毁。为了防止发生这种情况，用户可以将该窗口关键对象的 HandleVisibility 数

值暂时设置为 off：

```
user_input = get(editbox_handle,'String');
set(gui_handles,'HandleVisibility','off')
eval(user_input)
set(gui_handles,'HandleVisibility','commandline')
```

如果一个被保护的图形窗口是屏幕最顶层的窗口，而且在它之下存在未保护的窗口，那么使用 gcf 将返回最高层的未被保护的窗口，gca 的情况与之相同。如果不存在未保护的窗口或坐标轴，MATLAB 将创建一个窗口并返回它的句柄。

root 对象的 ShowHiddenHandles 属性是句柄可视的使能控制属性。ShowHiddenHandles 的缺省值为 off。当该属性值为 on 时，句柄对所有函数都是可视的。当用户希望访问某一特定时刻存在的所有对象时，该属性将十分有用。这里要说明的一点是，close 函数可以通过使用 hidden 选项来访问不可见的窗口，例如：

```
close('hidden')
```

即使该窗口是被保护的，该语句也将关闭屏幕最顶端的窗口。使用以下语句将关闭所有窗口：

```
close('all','hidden')
```

有些情况下需要改变坐标轴的可视外观来适应新的图形对象。为了与 MATLAB 高级函数的行为保持一致，在改变父坐标轴和图形窗口之前，最好先测试一下 hold 属性是否为 on，当 hole 属性为 on 时，坐标轴和图形窗口的 NextPlot 数值均为 add。以下的 my_plot3 函数将接受三维数据并使用 ishold 来检查 hole 属性的状态，以此来决定是否更改视图。

```
function my_plot3(x,y,z)
cax = newplot;
%检测当前的hold状态
hold_state = ishold;
LSO = ['- ';'--';': ';'-.'];
if nargin == 2
hlines = line(x,y,'Color','k');
%当且仅当hold为off时改变视图
if ~hold_state
view(2)
end
elseif nargin == 3
hlines = line(x,y,z,'Color','k');
% 当且仅当hold为off时改变视图
if ~hold_state
view(3)
end
end
ls = 1;
for hindex = 1:length(hlines)
```

```
if ls > length(LSO),ls = 1;end
set(hlines(hindex),'LineStyle',LSO(ls,:))
ls = ls + 1;
end
```

如果 hole 属性为 on，调用 my_plot3 时将不改变视图，否则如果有三个输入参数，那么 MATLAB 将视图由二维变为三维。

3. 关闭请求

定义 CloseRequestFcn 属性使用户能够阻止或延迟图形窗口的关闭以及 MATLAB 运行的终止。发生以下几种情况时，MATLAB 将执行由图形窗口的 CloseRequestFcn 属性定义的回调函数（称为关闭请求函数）：

- 在图形窗口中调用 close 命令；
- 用户退出 MATLAB 时还存在可见的图形窗口（如果一个窗口的 Visible 属性值为 off，则退出 MATLAB 时并不执行关闭请求函数，而是删除该图形窗口）；
- 使用窗口系统的关闭菜单或按钮来关闭图形窗口。

缺省的关闭请求函数保存在一个名为 closereq.m 的 M 文件中，该函数包括以下语句：

```
shh=get(0,'ShowHiddenHandles');
set(0,'ShowHiddenHandles','on');
delete(get(0,'CurrentFigure'));
set(0,'ShowHiddenHandles',shh);
```

这个关闭请求函数通过设置 root 对象的 ShowHiddenHandles 属性使得 HandleVisibility 属性的设置无效，即将所有图形窗口的句柄变为可见，这样，当用户退出 MATLAB 时，当前图形窗口的关闭请求函数将被调用，在关闭请求函数中删除该图形窗口，然后 root 对象 children 属性列表中的下一个窗口将变成当前窗口，执行它的关闭请求函数，如此重复该过程直至所有的窗口都被删除。如果用户修改了关闭请求函数使之不删除图形窗口（例如定义 CloseRequestFcn 属性值为空字符串），那么在图形窗口中发布关闭命令将不会导致图形窗口的删除。此时，由于 MATLAB 不删除图形窗口，所以 MATLAB 的退出也不能够被执行。

如果在试图关闭窗口时关闭请求函数发生错误，那么 MATLAB 将终止退出操作。但是如果使用 quit 或 exit 命令退出 MATLAB，即使关闭请求函数发生错误，MATLAB 也会无条件地关闭窗口。delete 函数也总是无视于 CloseRequestFcn 属性的取值而无条件地关闭指定的窗口，例如：

```
delete(get(0,'Children'))
```

将删除所有句柄可见的窗口（即所有 HandleVisibility 属性为 on 的窗口）。如果用户希望删除所有窗口（不论句柄是否隐藏），那么可以通过设置 root 对象的 ShowHiddenHandles 属性为 on 来实现。例如，以下语句：

```
set(0,'ShowHiddenHandles','yes')
delete(get(0,'Children'))
```

将无条件地删除所有图形窗口。

7.3.4　在 M 文件中保存句柄

所有句柄无论可见与否都能够保持其有效性，如果用户确实知道某对象的句柄，无论该句柄可见与否，用户都可以使用句柄进行对象属性的设置或查询。

图形 M 文件经常使用句柄来访问属性值，并通过句柄直接定义图形输出的目标。MATLAB 提供一些有用的函数来获得图形关键对象（例如当前窗口和坐标轴）的句柄，然而在 M 文件中，使用这些函数并不是获得句柄最好的方法。这是因为，在 MATLAB 中查询对象句柄或其他信息的执行效率不高，最好还是将句柄直接保存为一个变量进行引用。另外，由于当前的坐标轴、图形窗口或对象有可能因为用户的交互而发生变化，查询方式难以确保句柄完全正确，而使用句柄变量则可以保证能够体现对象发生的变化。

为了保存句柄信息，通常在用户 M 文件的开始处保存 MATLAB 相关的状态信息。例如，用户可以使用以下语句作为 M 文件的开头：

```
cax = newplot;
cfig = get(cax,'Parent');
hold_state = ishold;
```

这样就无需在每次需要这些信息时都重新进行查询。如果用户在 M 文件中暂时改变了保持状态，用户应当将 NextPlot 的当前属性值保存下来，以便以后重新设置。

```
ax_nextplot = lower(get(cax,'NextPlot'));
fig_nextplot = lower(get(cfig,'NextPlot'));
set(cax,'NextPlot',ax_nextplot)
set(cfig,'NextPlot',fig_nextplot)
```

有一些内置的函数可以修改坐标轴的属性以实现某种特定的效果，这些函数可能会对用户的 M 文件产生一定的影响。

表 7-3 列出了一些 MATLAB 的内置函数以及它们所修改的属性。注意这些属性仅在 hold 置为 off 时才会发生变化。

表 7-3　某些 MATLAB 的内置函数修改的属性

函数名	坐标轴属性（改变后）	函数名	坐标轴属性（改变后）
fill	Box: on CameraPosition:2-D view CameraTarget:2-D view CameraViewAngle:2-D view CameraUpVector:2-D view	plot	Box: on CameraPosition:2-D view CameraTarget:2-D view CameraViewAngle:2-D view CameraUpVector:2-D view
fill3	CameraPosition:3-D view CameraTarget:3-D view CameraUpVector:3-D view CameraViewAngle:3-D view XScale: linear YScale: linear ZScale: linear	plot3	CameraPosition:3-D view CameraTarget:3-D view CameraUpVector:3-D view CameraViewAngle:3-D view XScale: linear YScale: linear ZScale: linear

（续表）

函数名	坐标轴属性（改变后）	函数名	坐标轴属性（改变后）
image (high-level)	Box: on Layer: top CameraPosition:2-D view CameraTarget:2-D view CameraUpVector:2-D view CameraViewAngle:2-D view XDir: normal XLim: [0 size(CData,1)]+0.5 XLimMode: manual YDir: reverse YLim: [0 size(CData,2)]+0.5 YLimMode: manual	semilogx	Box: on CameraPosition:2-D view CameraTarget:2-D view CameraUpVector:2-D view CameraViewAngle:2-D view XScale: log YScale: linear
loglog	loglog Box: on CameraPosition:2-D view CameraTarget:2-D view CameraUpVector:2-D view CameraViewAngle:2-D view XScale: log YScale: log	semilogy	Box: on CameraPosition:2-D view CameraTarget:2-D view CameraViewAngle:2-D view CameraUpVector:2-D view XScale: linear YScale: log

7.4 实 例 讲 解

【实例】：如何设置坐标轴的调色板来改变图形输出效果？

分析：在本问题中将向读者介绍如何设置坐标轴的自定义属性，读者可以据此来了解具体地定义某种特殊图形对象的属性。坐标轴是一幅图形中最为重要的对象，通过设置其属性可以获得许多种不同的显示效果。读者可以从 axis.m 文件中获得能够设置的坐标轴属性的种类。这里仅介绍其调色板范围属性 CLim 的设置方法。

CLim 控制图像、面片和曲面的颜色数据 CData 到图形窗口调色板的映射方法。CLim 是一个二元素的向量[cmin，cmax]，它将 CData 的最小值映射到窗口调色板的第一项(cmin)，将最大值映射到调色板最后一项（cmax），其他数据按照以下公式映射：

调色板索引= 取整{ (Cdata-cmin)/(cmax-cmin)*cm_length }+1

可见为了计算 CLim 的取值，需要了解并设置以下几个变量：调色板总长 CmLength、每一个坐标轴使用的调色板入口 BeginSlot 和调色板出口 EndSlot 以及 CData 的最大最小值。

本例中将使用两个不同的调色板对同一组地形数据进行绘制，第一幅图形使用普通的调色方法（绿色为陆地，蓝色为海洋），调色板名称为 Lightingmap；第二幅图形使用一种从暗到明的三维照明着色方法，调色板名称为 Atlasmap。为了实现这一目的，将两幅图形所需的调色板联合起来，不同的图形将使用联合调色板的不同部分。为了实现调色板的联合，要计算联合后的 CLim 属性值并将该属性值赋给每一个坐标轴。

解决方法：

步骤一：设置绘图区域并绘制曲面：

```
ax1 = subplot(2,1,1);
view([0 80])
```

```
        surf(topodata)
        shading interp
        ax2 = subplot(2,1,2),;
        view([0 80]);
        surfl(topodata,[60 0])
    shading interp
```

步骤二：连接两个系统调色板形成新的调色板并装载：

```
    colormap([Lightingmap;Atlasmap]);
```

步骤三：获取计算新的 CLim 所需的数据：

```
        CmLength = size(get(gcf,'Colormap'),1);
        BeginSlot1 = 1;
        EndSlot1 = size(Lightingmap,1);
        BeginSlot2 = EndSlot1+1;
        EndSlot2 = CmLength;
        CLim1 = get(ax1,'CLim');
    CLim2 = get(ax2,'CLim');
```

步骤四：创建一个计算 CLim 属性值的函数：

```
    function CLim = newclim(BeginSlot,EndSlot,CDmin,CDmax,CmLength)
        PBeginSlot = (BeginSlot -1) / (CmLength-1);
        PEndSlot = (EndSlot -1) / (CmLength-1);
        PCmRange = PEndSlot -PBeginSlot;
        DataRange = CDmax -CDmin;
        ClimRange = DataRange / PCmRange;
        NewCmin = CDmin -(PBeginSlot * ClimRange);
    NewCmax = CDmax + (1-PEndSlot) * ClimRange;
    CLim = [NewCmin,NewCmax];
```

步骤五：使用该函数计算 CLim 属性值，更新每个坐标轴的 CLim 属性，使第一个坐标轴使用联合调色板的 65 到 120 项，第二个坐标轴使用联合调色板的 1 到 64 项：

```
    set(ax1,'CLim',newclim(65,120,clim1(1),clim1(2)))
    set(ax2,'CLim',newclim(1,64,clim1(1),clim1(2)))
```

显示结果分别如图 7-12、图 7-13 所示。



图 7-12　使用调色板 65 到 120 项的显示效果



图 7-13　使用调色板 1 到 64 项的显示效果

7.5 小　　结

本章主要向读者介绍了有关图形对象、对象属性和句柄的概念及使用方法。通过本章的学习，读者将会掌握创建图形的低级方法，这不但对理解和掌握 MATLAB 高级绘图函数有所帮助，而且为以后创建和使用用户图形界面（GUI）打下了基础。

第 8 章　在 MATLAB 中创建图形用户接口

一个可发布的应用程序通常都需要具备一个友好的图形界面，本章就来介绍创建图形用户界面（GUI）的具体方法。本章首先对 GUI 的基本概念作简单的介绍，然后说明 GUI 开发环境 GUIDE 及其组成部分的用途和使用方法，最后说明创建一个完整 GUI 的详细步骤。

8.1　图形用户界面概述

8.1.1　GUI 开发方法简介

通常在开发一个实际的应用程序时都会尽量做到界面友好，最为常用的方法就是使用图形界面。提供图形用户界面的应用程序能够使用户的学习和使用更为方便容易。用户不需要知道应用程序究竟是怎样执行各种命令的，而只需要了解可见界面组件的使用方法；用户也不需要知道命令是怎样执行的，只要通过与界面交互就可以使指定的行为得以正确执行。

在 MATLAB 中，图形用户界面是一种包含多种对象的图形窗口。用户必须对每一个对象进行界面布局和编程，从而使用户激活 GUI 每个对象时都能够执行相应的行为。另外，用户必须保存和发布所创建的 GUI，使得 GUI 能够真正地得到应用。

MATLAB 为用户开发图形界面提供了一个方便高效的集成环境：MATLAB 图形用户界面开发环境 GUIDE（MATLAB's Graphical User Interface Development Environment）。上述所有工作都能够使用 GUIDE 方便地实现。GUIDE 主要是一个界面设计工具集，MATLAB 将所有 GUI 支持的用户控件都集成在这个环境中并提供界面外观、属性和行为响应方式的设置方法。GUIDE 将用户保存设计好的 GUI 界面保存在一个 FIG 资源文件中，同时还能够生成包含 GUI 初始化和组件界面布局控制代码的 M 文件。这个 M 文件为实现回调函数（当用户激活 GUI 某一组件时执行的函数）提供了一个参考框架。虽然使用用户自己编写的、包含 GUI 所有发布命令的 M 文件也能够实现一个 GUI，但是使用 GUIDE 执行效率更高；使用 GUIDE 不但能够交互式地进行组件界面布局，而且能够生成两个用来保存和发布 GUI 的文件：

- FIG 文件：该文件包括 GUI 图形窗口及其所有后裔的完全描述，包括所有对象的属性值。FIG 文件是一个二进制文件，调用 hgsave 命令或界面设计编辑器的 File 菜单 Save 选项保存图形窗口时将产生该文件。FIG 文件包含序列化的图形窗口对象，在用户打开 GUI 时，MATLAB 能够通过读取 FIG 文件重新构造图形窗口及其所有后裔。所有对象的属性都被设置为图形窗口创建时保存的属性。缺省情况下，即使用

户使用 hgsave 和 hgload 命令保存系统缺省的图形工具条和菜单，FIG 文件也不保存这些缺省信息。FIG 文件最有用的地方之一就是对象句柄的保存和引用。可以使用 open、openfig 和 hgload 命令来打开一个后缀为.fig 的文件；

- M 文件：该文件包括 GUI 设计、控制函数以及定义为子函数的用户控件回调函数，主要用于控制 GUI 展开时的各种特征。这个 M 文件可分为 GUI 初始化和回调函数两个部分，用户控件的回调函数根据用户与 GUI 的具体交互行为分别调用。这里将 GUI 的 M 文件称为应用程序 M 文件。应用程序 M 文件使用 openfig 命令来显示 GUI。注意应用程序 M 文件并不包括用户界面设计的任何代码，这些代码将完全由 FIG 文件保存。

GUIDE 可以根据用户 GUI 的版面设计过程直接自动生成 M 文件框架，这样就简化了 GUI 应用程序的创建工作，用户可以直接使用这个框架来编写自己的函数代码。这样的编写方法具有以下优点：

- 应用程序 M 文件已经包含实现一些有用的函数编写代码，无需用户自行编写；
- 可以使用该 M 文件生成的有效方法来管理图形对象句柄并执行回调函数子程序；
- 提供管理全局数据的途径；
- 文件支持自动插入回调函数原型，确保当前 GUI 与未来发布版本的兼容性。

用户也可以选择由 GUIDE 生成 FIG 文件、自己编写应用程序 M 文件的 GUI 创建方式。编写 M 文件时要注意，应用程序 M 文件中不能包含用户控件创建命令，所有的界面设计信息都保存在由界面设计编辑器生成的 FIG 文件中。

实现一个 GUI 主要包括以下两项工作：GUI 界面设计和 GUI 组件编程。整个 GUI 的实现过程可以分为以下几步：

- 通过设置 GUIDE 应用程序的选项来进行 GUIDE 组态；
- 使用界面设计编辑器进行 GUI 界面设计；
- 理解应用程序 M 文件中所使用的编程技术；
- 编写用户 GUI 组件行为响应控制（即回调函数）代码。

8.1.2 GUIDE 支持的组件类型

界面设计编辑器组件平台中包含所有能够在 GUI 中使用的用户界面控件。这些控件都属于 MATLAB 的用户控件对象（uicontrol），可以通过 Callback 属性来进行回调函数编程。下面将简单介绍各种控件的概念和特点：

- 按钮：通过鼠标点击按钮可以实现某种行为（按钮陷下和弹起等）并调用相应的回调子函数；
- 拴牢按钮：拴牢按钮能够产生一个二进制状态的行动（on 或 off）。点击该按钮将使按钮的外观保持陷下状态，同时调用相应的回调函数。再次点击该按钮将使按钮弹起，同样也要调用回调函数。拴牢按钮的回调函数首先要对按钮的状态进行查询，然后才能决定相应的行为；
- 单选按钮：单选按钮与按钮的执行方式没有本质上的区别，但是单选按钮通常以组为单位，一组单选按钮之间是一种互相排斥的关系，也就是说任何时候一组单选按

钮中只能有一个有效；

- 复选框：复选框与单选按钮类似，只是多个复选框可以同时有效。复选框为用户提供一些可以独立选择的选项进行设置程序模式，例如显示工具条与否以及生成回调函数原型与否等等；
- 编辑框：编辑框是控制用户编辑或修改字符串的文本域，其 String 属性包含用户输入的文本信息。如果一个编辑框有输入焦点，在 UNIX 系统中，点击图形窗口的菜单栏，编辑框回调函数就会被调用，而在 Windows 系统下，点击图形窗口不会导致编辑框回调函数的执行。这个区别是出于操作平台方便性考虑而定的；
- 静态文本：静态文本通常作为其他控件的标签使用，用户不能采用交互方式修改静态文本或调用相应的回调函数；
- 滚动条：使用户能够通过移动滚动条来改变指定范围内的数值输入，滚动条的位置代表用户输入的数值；
- 组合框：组合框是图形窗口中的一个封闭区域，它把相关联的控件（例如一组单选按钮）组合在一起，使得用户界面更容易理解；
- 列表框：列表框显示由其 String 属性定义的一系列列表项并使用户能够选择其中的一项或多项；
- 弹出式菜单：弹出式菜单将打开并显示一个由其 String 属性定义的选项列表。当用户希望提供一些相互排斥的选项但不希望使用一系列占用有限空间的单选按钮时，文本菜单将非常有用。

GUI 中可以存在一个或多个以上的 GUI 组件，使用时要注意保证各个组件的名称或属性有所不同，可以区分。

8.2　GUIDE 及其组成部分

GUIDE 提供一个界面设计工具集实现图形用户界面的创建工作。这些工具包括：

- 界面设计编辑器：添加并排列图形窗口中的组件对象；
- 属性检查器：检查并设置组件的属性值；
- 对象浏览器：观察此次 MATLAB 运行过程中图形对象的句柄继承关系表；
- 菜单编辑器：创建窗口菜单和文本菜单。

以上这些工具通过界面设计编辑器的相应菜单进行调用。使用 guide 命令来启动界面设计编辑器：

```
guide
```

界面设计编辑器的外观如图 8-1 所示。

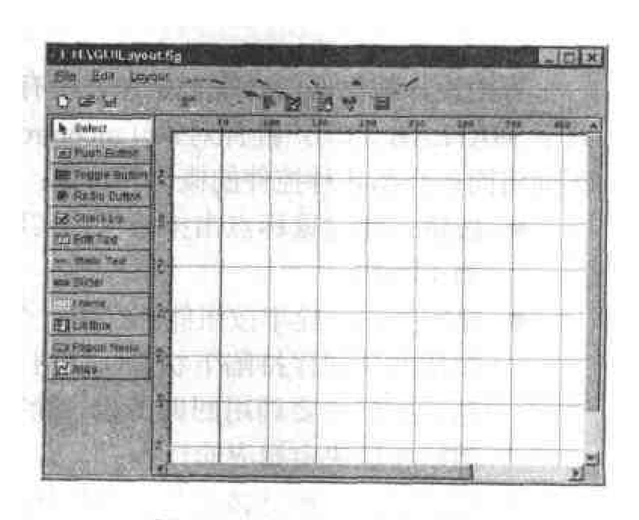


图 8-1　界面设计编辑器外观

8.2.1 GUI 设计——界面设计编辑器

界面设计编辑器使用户能够从组件面板中选择 GUI 组件并将它们排列在图形窗口中。界面设计编辑器由四个部分组成：组件面板、工具栏、菜单栏和界面区域。组件面板包含用户界面可获得的所有组件（用户控件对象）；工具栏和菜单栏可以用来启动其他界面设计工具，例如菜单编辑器；界面区域实际上就是激活后的 GUI 图形窗口。

1. 组件面板

在 GUI 界面中放置组件的方法和步骤是这样的：首先点击组件面板的相应按钮，选择用户希望放置的组件类型，光标变为十字形后使用十字形光标的中心点来确定组件右上角的位置，或者通过在界面区域内点击并拖动鼠标来确定组件的大小。图 8-2 给出了一个添加组件的示例。

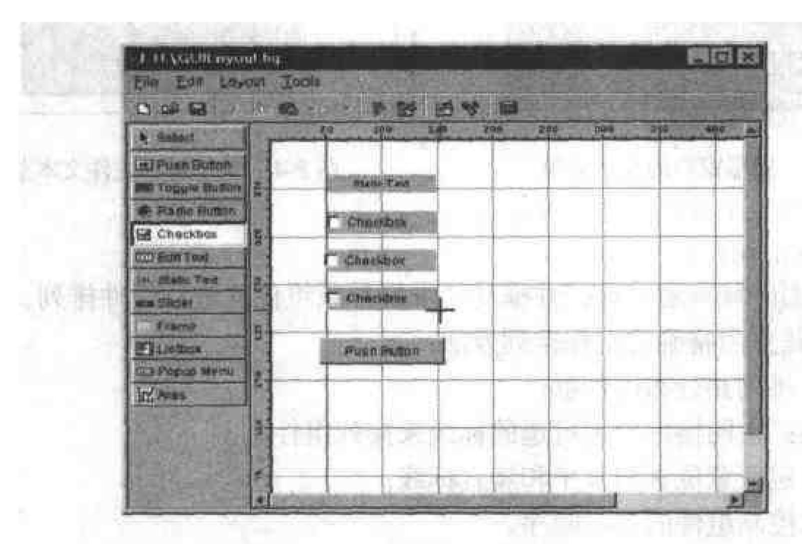


图 8-2 在界面中添加组件的示例

GUI 组件布置工作完成后，用户可以使用激活按钮或选择 Tools 菜单的 Activate Figure 选项来观察 GUIDE 的设计结果。激活图形窗口将发生以下事件：首先保存 FIG 文件和 M 文件，如果用户尚未保存该设计结果，GUIDE 将打开保存对话框使用户选择将要创建的 M 文件名；然后 GUIDE 保存与 M 文件同名的 FIG 文件（扩展名为.fig），如果存在一个同名的 M 文件，GUIDE 将会显示一个提示对话框，如果用户选择提示对话框的 Replace 按钮，原来的 M 文件将被替换；如果选择 Append，则 GUIDE 将向原有的 M 文件中插入未保存的新组件回调函数并根据应用程序选项对话框设置的变化来修改原有代码。注意如果 FIG 文件和 M 文件已经存在，那么当用户激活 GUI 时，GUIDE 将自动保存应用程序 M 文件和 FIG 文件，因此不要随意激活 GUI 以免有用的组件被替换。

2. 文本菜单

使用界面设计编辑器进行界面设计时，可以使用鼠标左键来选择一个对象，然后点击右键来显示与所选对象相连的文本菜单。图 8-3 表示了一个与图形窗口对象联系在一起的文本菜单，所有已定义的回调子函数都列举在该菜单的下方。

图 8-4 描述了一个与按钮相联系的文本菜单，同样，所有已定义的回调函数都列举在该

菜单的下方。

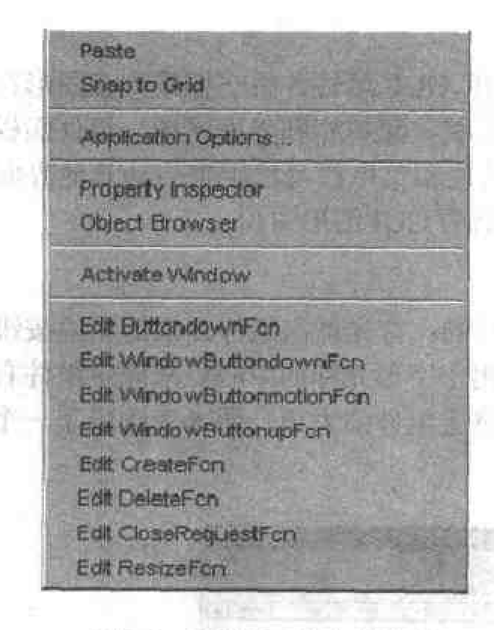


图 8-3　图形窗口的文本菜单

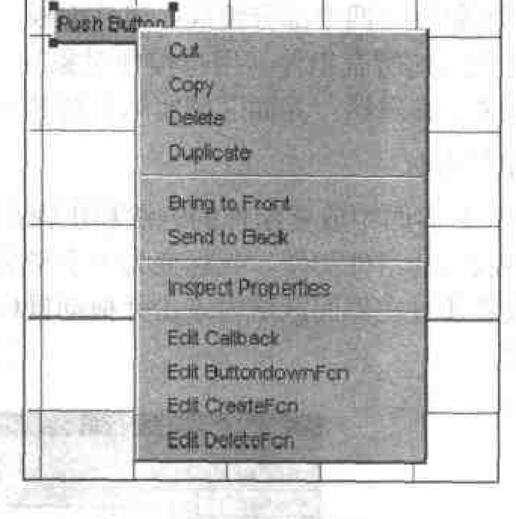


图 8-4　GUI 用户控件文本菜单

3. 排列工具

用户可以在界面区域内通过选择并拖动任意组件或组件群进行组件排列。另外，界面设计编辑器还提供一些更为精确的组件排列方法：

- 排列工具：排列并分布组件群；
- 栅格和标尺：在网格内使用可选的标线来排列组件；
- 指引线：指定任意位置的水平和竖直标线；
- 拉前推后：控制组件的前后顺序。

排列工具使用户能够根据其他组件的位置调整被选择物体的间隔或放置某个组件。上述的排列方法可以通过排列工具栏获得，当用户按下 Apply 按钮时，指定的排列操作将应用于所有被选择的组件。图 8-5 显示了排列工具栏的外观。

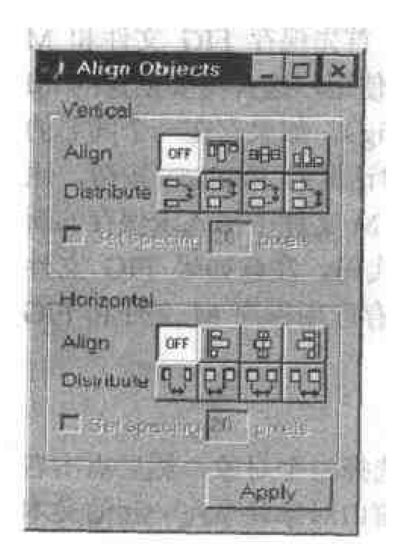


图 8-5　排列工具栏外观

排列工具提供两种类型的排列操作：

- Align：按照一个单参考线来排列所有被选择的组件；
- Distribute：根据组件间的关系进行所有被选组件的统一放置。

这两种排列操作都能够在水平和竖直两个方向上使用。最好使用两个步骤各自独立地完成水平和竖直方向上的排列工作。

第一种排列操作方式按照一条水平或竖直参考线来排列由范围框决定的被选对象群。图 8-6 的界面区域图形显示了一个由三个被选择的按钮组成的范围框（用虚线表示）。所有的对齐操作都根据范围框的边界来执行，组件对齐方向可以是水平上方、中间和底部以及竖直左边、中间和右边。

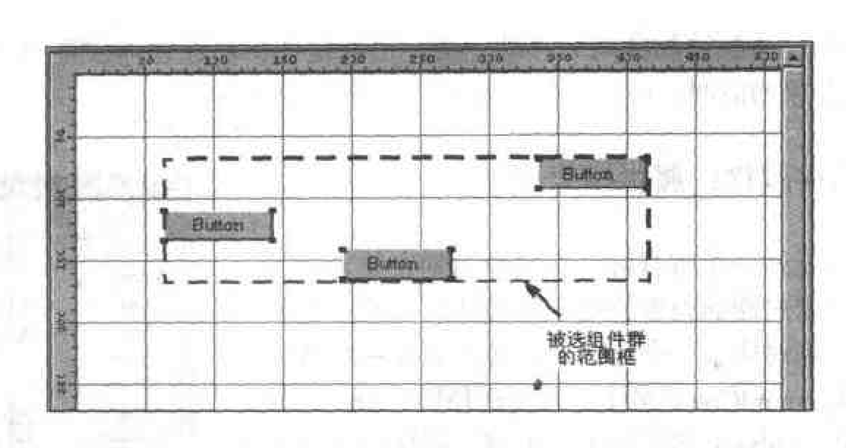


图 8-6　被选组件群的范围框

第二种排列方法可以为所选组件群的所有组件添加相等的间隔。这种排列方式可以按照两种不同的模式进行操作：在范围框内以相等间隔分布所选的组件；通过点击 Set spacing 指定像素值来分布所选的组件。这两种模式都可以指定控件间隔的度量方式，常用的间隔度量方式是按照几种边界进行的：

- 竖直：内心、顶端、中心和底部；
- 水平：内心、左边、中间和右边。

4. 网格和标线

界面区域可以使用网格和标线辅助完成组件设计工作。用户可以将网格线的间隔设置在 10 到 200 个像素之间，缺省情况下以 50 个像素为间隔。如果用户选择了 snap-to-grid 选项，那么对于任何一个在网格线周围 9 像素范围内移动或重画的对象，系统都会自动将该对象放置在该网格线上。无论网格是否可见，snap-to-line 选项都是有效的。

网格和标线对话框外观如图 8-7 所示。

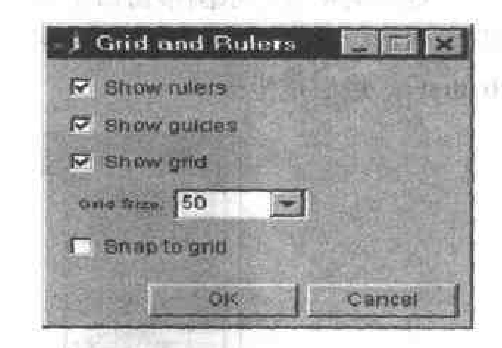


图 8-7　网格和标线对话框外观

该对话框可以完成以下工作：

- 控制标线、网格和指引线的可见与否；
- 设置网格间距；
- 使能 snap-to-line。

界面设计编辑器有水平和竖直两种指引线。当用户希望在界面设计编辑器的任意位置建立一个组件排列参考标准时，指引线将非常有用。点击标线的左边或顶端并将其拖放到界面区域中所需位置处就会生成一条指引线。

界面设计编辑器提供四种控制交叠对象前后关系的操作：

- 放到最前：将被选物体放置到未选中对象的前面，可以通过文本菜单或快捷键 Ctrl+F 实现；
- 放到最后：将被选物体放置到未选中对象的后面，可以通过文本菜单或快捷键 Ctrl+B 实现；
- 向前移动：将被选物体向前移动一级，也就是说将被选物体放置到与该对象相连的高一级对象的前面，可以从 Layout 菜单中获得该操作；

- 向后移动：将被选物体向后移动一级，也就是说将被选物体放置到与该对象相连的低一级对象的后面，可以从 Layout 菜单中获得该操作。

8.2.2 设置组件属性：属性检查器

属性检查器提供一个所有可设置属性的列表并显示当前的属性值，使用户能够设置界面中各组件的属性。列表中的每一个属性都对应于一个相应于该属性的属性值选择范围，例如 BackgroundColor 属性的颜色选择器、FontAngle 的弹出式菜单和 Callback 字符串的文本域。属性检查器的外观如图 8-8 所示。

用户可以通过多种方式来启动属性检查器：双击界面设计编辑器中的组件；选择 Tools 菜单下的 Property Inspector 选项；选择 Edit 菜单下的 Inspect Property 选项；从组件的文本菜单中选择 Inspect Property 选项。

通过对 GUI 中各个用户控件对象的属性设置可以实现用户所需的控件外观和行为特征。

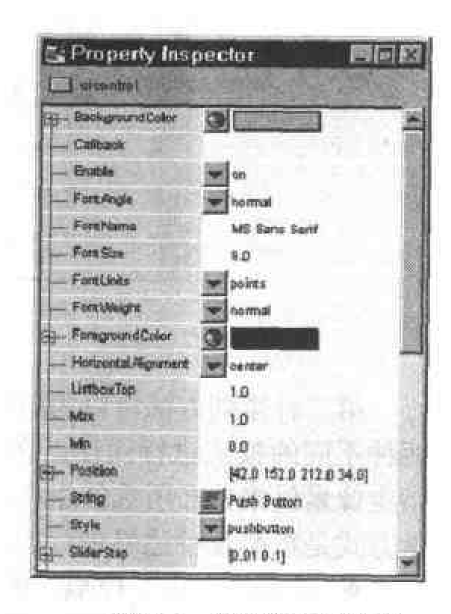


图 8-8 属性检查器外观

8.2.3 观察对象继承表：对象浏览器

对象浏览器显示图形窗口中所有对象的继承关系。在图 8-9 给出的 GUI 中，从对象浏览器中可以看出，第一个创建的用户控件是组合框(frame)，然后是 4 个单选按钮（Radio button），最后是坐标轴。

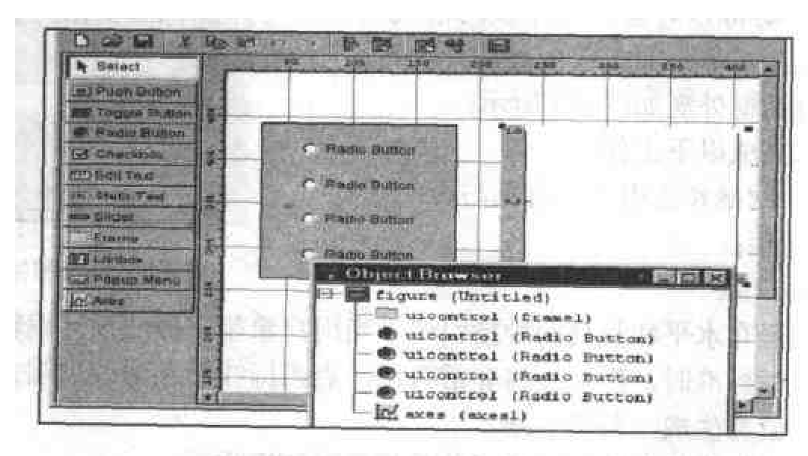


图 8-9 对象浏览器范例

8.2.4 创建菜单：菜单编辑器

菜单编辑器的外观如图 8-10 所示。

GUIDE 能够创建两种类型的菜单：在图形窗口菜单栏中显示的菜单栏菜单；当用户右击图形对象时弹出的文本菜单。可以使用菜单编辑器来创建这两种类型的菜单。

1. 定义菜单栏菜单

创建菜单的第一步是使用New Menu工具栏来创建一个菜单，然后来指定菜单的属性。用户点击创建的菜单项将会显示如图8-11所示的一个文本域，在该文本域中可以设置菜单的标签、分隔符、选中模式以及回调函数字符串。

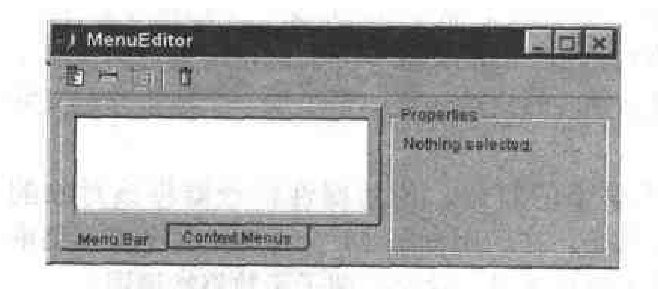


图8-10 菜单编辑器外观

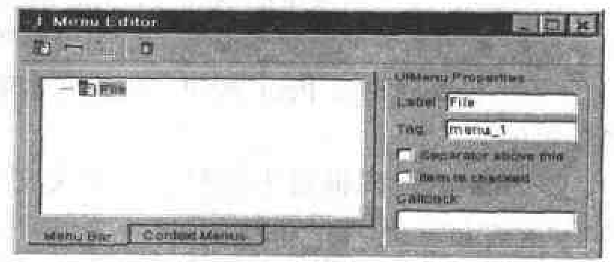


图8-11 菜单的相关设置

菜单创建成功后，MATLAB将该菜单添加到图形窗口的菜单栏中。

下面进行第二个步骤：创建菜单项。使用New Menu Item工具来添加菜单项，每一个菜单项也可以有级联的子菜单项。图8-12显示了为窗口菜单栏定义了三个菜单的菜单编辑器外观。

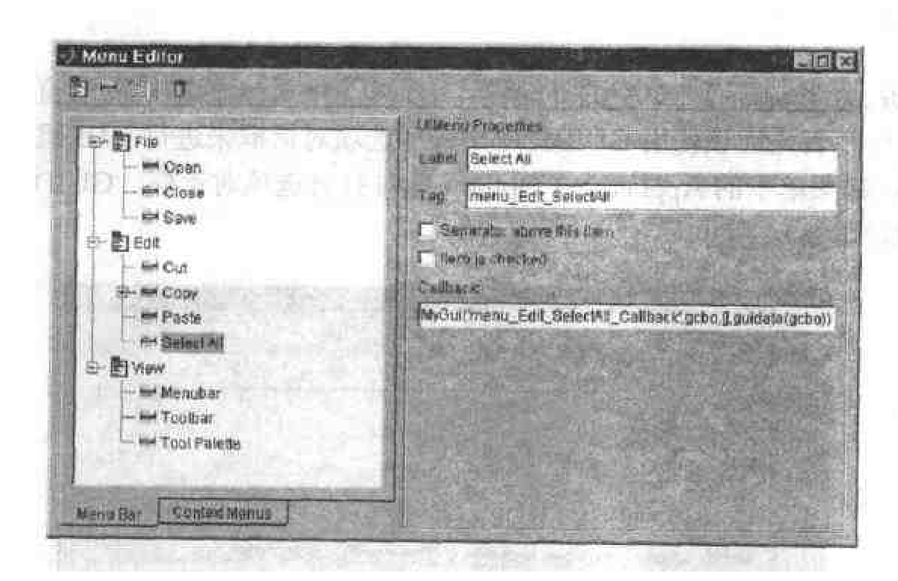


图8-12 菜单项创建示例

如果用户激活图形窗口，这三个菜单将会出现在窗口中（如图8-13所示）。



图8-13 菜单创建结果

2. 定义文本菜单

定义了文本菜单的对象后，当用户右击鼠标时，文本菜单随之出现。菜单编辑器能够定义文本菜单并将菜单与对象联系起来。

文本菜单的所有菜单项都是文本菜单的子对象，这些菜单项并不显示在窗口菜单栏中。选择菜单编辑器工具条中的 New Context Menu 来创建父菜单并为之定义一个标签（名称）。注意在定义文本菜单之前要选择菜单编辑器的 Context Menus 标签界面。使用菜单编辑器工具条中的 New Menu Item 按钮来创建文本菜单项，然后给该菜单项添加一个标签并定义回调字符串。

在界面设计编辑器中选择需要定义文本菜单的对象，使用属性检查器将该对象的 UIContextMenu 属性设置为所需文本菜单的标签名。在应用程序 M 文件中给每个文本菜单项添加一个回调子函数，当用户选择特定的文本菜单项时，这个回调子函数将被调用。

8.3　使用 GUIDE 创建 GUI

8.3.1　GUI 组态

调用 guide 命令来显示一个空的界面设计编辑器和一个无标题的图形窗口。在为该图形窗口添加组件时，首先应该使用 GUIDE 应用程序选项对话框来进行 GUI 组态。选择界面设计编辑器的 Tools 菜单下的 Application Options 选项打开选项对话框。GUIDE 应用程序选项对话框的外观如图 8-14 所示。

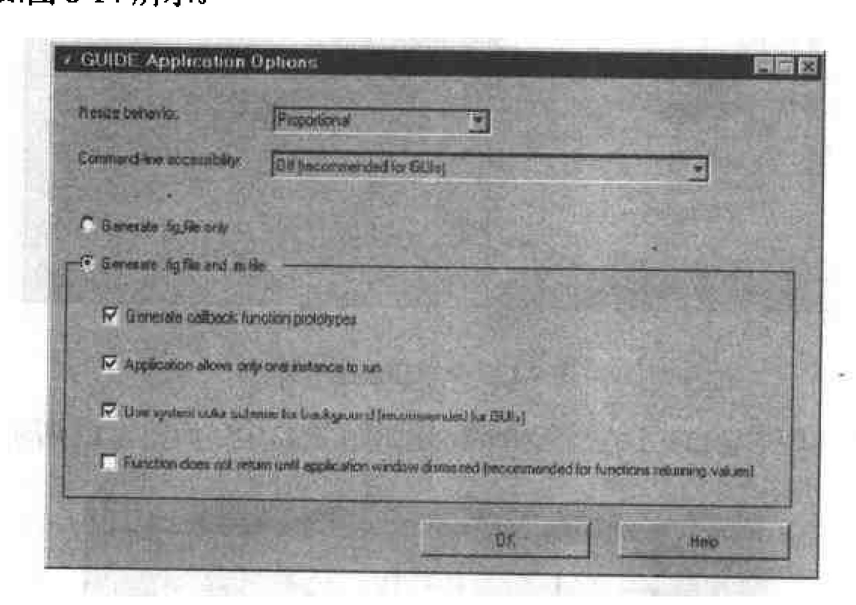


图 8-14　GUIDE 应用程序选项对话框

该对话框能够设置的选项包括以下几项：

- 窗口重画行为（Resize behavior）；
- 命令行访问（Command-line accessibility）；

- 仅生成 FIG 文件/生成 FIG 文件和 M 文件（General.fig file only/General.fig file and .m file）；
- 生成回调函数原型（General Callback function prototypes）；
- 同一时刻仅允许运行一个应用程序实例（Applications allows only one instance to run）；
- 使用系统背景颜色配置（Use system color for background (recommended for GUIs)）；
- 函数直到应用程序窗口销毁才返回（Function does not return until application window dismissed(recommended for functions returning values)）。

1. 重画行为（Resize behavior）

通过选项对话框可以决定用户是否可以重画 GUI 所在的图形窗口以及 MATLAB 将如何管理重画过程。GUIDE 提供以下三种选择：

（1）Non-resizable：用户不能改变窗口大小（缺省选项）。某些类型的 GUI 是不能够重画的，例如警告和简单询问对话框，其交互行为非常简单，没有必要实现重画。GUIDE 将设置以下属性值来实现这种类型的 GUI：

- units：图形窗口、坐标轴和用户控件的 units 属性应该被设置为 characters；
- Resize：图形窗口的 resize 属性将被设置为 off；
- ResizeFcn：不给图形窗口的 ResizeFcn 属性定义回调函数。

（2）Proportional：允许 MATLAB 按照新的图形窗口尺寸自动按比例重新绘制 GUI 组件。注意重画过程将不改变组件标签字体的大小，如果图像窗口过大，那么组件的标签将不可读。对于在设置过程中始终不关闭的简单 GUI 工具和对话框来说，这种方式非常实用。GUIDE 将设置以下属性值来实现这种类型的 GUI：

- units：图形窗口 units 属性应该被设置为 characters：坐标轴和用户控件的 units 属性应该被设置为 normalized；
- Resize：图形窗口的 resize 属性将被设置为 on；
- ResizeFcn：不给图形窗口的 ResizeFcn 属性定义回调函数。

（3）User-specified：通过编程使重画过程中 GUI 按照用户指定的方式变化。该选项需要用户编写一个 ResizeFcn 属性定义的回调函数，该函数根据新的图形窗口尺寸重新计算组件的大小和位置。通过对图形窗口的 ResizeFcn 回调函数进行编程，用户可以创建一个适合重画并保持原始设计外观的 GUI。这种方式适合于需要在运行过程中与用户进行交互的 GUI，因为这样的应用程序可能会包括一些数据坐标与界面相矛盾的组件。GUIDE 使用以下属性值实现这种类型的 GUI：

- units：图形窗口、坐标轴和用户控件的 units 属性应该被设置为 characters；
- Resize：图形窗口的 resize 属性将被设置为 on；
- ResizeFcn：图形窗口的 ResizeFcn 属性需要一个控制重画的回调函数。

2. 命令行访问（Command-line accessibility）

当 MATLAB 创建一幅图形时，该图形的图形窗口和坐标轴将包括在其父对象的子对象列表中，它们的句柄可以使用诸如 findobj、set 和 get 之类的函数来获得。创建 GUI 时也要创建一个图形窗口，由于绘图命令会直接将结果输出到当前的 GUI 图形窗口中，所以通常情况下用户不需要额外的 GUI 图形窗口访问就可以直接将图形输出到窗口中。但是，如果

用户需要创建一个包含坐标轴等绘图工具的 GUI，那么访问图形窗口则是必要的。这种情况下 GUI 要支持命令行的访问。

GUIDE 应用程序选项对话框提供三种用户访问权限：

- off：禁止命令行对 GUI 图形窗口的访问；在这种方式下 GUI 图形句柄是隐藏的，这就意味着用户不能够使用 findobj 函数来定位 GUI 中的用户控件句柄。应用程序 M 文件将创建一个对象句柄结构体来保存 GUI 中的所有用户控件句柄并将该结构体传递给子函数保证 GUI 中句柄的使用无误；
- On：允许命令行对 GUI 图形窗口进行访问；
- User-specified：GUI 使用用户设置的图形窗口 HandleVisibility 和 IntegerHandle 属性值，这两个属性决定句柄是否能被命令行获得。HandleVisibility 属性决定图形窗口的句柄对试图访问当前图形窗口的命令是否可见：该属性设置为 off 将导致图形窗口的句柄从根对象的子对象列表中删除，使该图形窗口不再是当前图形窗口。但是该图形窗口的句柄仍然有效，明确使用该句柄的函数仍然能够工作，例如，如果图形窗口句柄为 1，则 colse(1)仍然能够执行；IntegerHandle 决定图形窗口的句柄是一个整数还是浮点数。如果该属性为 off，MATLAB 将使用一个不可重复使用的浮点数（例如 67.0001221）来代替整数，这将大大减小用户对图形窗口的误操作机率。

3. 选择仅生成 FIG 文件（General.fig file only）

如果用户不希望 GUIDE 生成应用程序 M 文件，在 DUIDE 应用程序选项对话框中选择 Generate .fig file only 选项。当用户在界面设计编辑器中保存 GUI 时，GUIDE 将仅仅创建能够使用 open 或 hgload 命令重新显示的 FIG 文件。当用户选择这个选项时，必须将每一个 GUI 组件的 Callback 属性设置为一个 MATLAB 能够理解并执行指定操作的字符串，这个字符串可以是一个表达式也可以是一个 M 文件。如果用户希望生成一个与应用程序 M 文件完全不同的程序范例时可以选择这个选项。

如果用户希望 GUIDE 同时创建 FIG 文件和应用程序 M 文件，在 GUIDE 应用程序选项对话框中选择 Generate .fig file and .m file 选项。一旦用户选择了这个选项，用户就可以选择以下任何一项 M 文件的组态项目：

- 生成回调函数原型（Generate callback function prototypes）：当用户在 GUIDE 应用程序选项对话框中选择该选项时，GUIDE 将在应用程序 M 文件中为每一个组件添加一个回调子函数（注意组合框和静态文本组件不使用自身的 Callback 属性）。用户必须为回调函数编写代码。GUIDE 还为用户添加一个与弹出式（右击鼠标出现）文本菜单选项对应的回调子函数，用户无论何时选择菜单项都将调用该函数。回调函数的语法格式如下：function objectTag_Callback(h,eventdata,handles,varargin)。函数参数的意义如表 8-1 所示。

表 8-1　回调函数参数名称及含义

参 数 名	参 数 含 义
h	回调函数被调用对象的句柄
eventdata	保留，为空
handles	包含 GUI 中所有组件句柄的结构体，该结构体的域名由对象的 Tag 属性定义。也可以用来传递数据给其他回调函数和主程序。
varargin	希望传递给回调函数的参数变量长度列表

例如，假设创建一个包含按钮组件的 GUI，该按钮的 Tag 属性被设置 pushbutton1，GUIDE 在应用程序 M 文件中生成如下的回调子函数：

function pushbutton1_Callback(h,eventdata,handles,varargin)

然后设置该按钮的 Callbace 属性：

mygui('pushbutton1_Callback',gcbo,[],guidata(gcbo))

其中：

mygui：该 GUI 的 FIG 文件名；

pushbutton1_callback：回调子函数名；

gcbo：返回按钮句柄；

[ ]：为 eventdata（事件数据）参数保留的空矩阵；

guidata(gabo)：从图形窗口的应用程序数据中获得句柄结构体。

如果用户希望将其他参数传递给按钮的回调子函数，使用属性检察器给 Callback 属性添加一个由逗号分隔的参数列表（该列表由子函数中的 varargin 参数控制）。例如，如果用户希望在名为 MyGui.m 的应用程序 M 文件中给按钮回调函数添加两个参数，需要对 M 文件做以下修改：

首先要编辑回调子函数的定义行以添加参数：

function varargout = pushbutton1_Callback(h,eventdata,handles,arg1,arg2)

然后编辑按钮回调子函数以添加参数。编辑界面如图 8-15 所示。

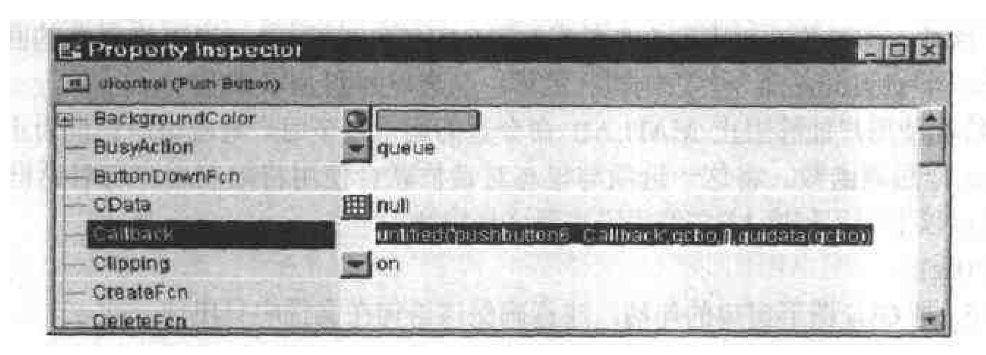


图 8-15　按钮回调子函数编辑界面

- 仅允许应用程序的一个实例运行（Application Allows Only One Instance to Run）：该选项允许用户选择图形窗口的以下两种行为：允许 MATLAB 的一次运行过程中仅有 GUI 的一个实例；允许 MATLAB 显示 GUI 的多个实例。

如果用户仅允许运行一个实例，那么无论何时执行一个发布 GUI 的命令 MATLAB 都将重新使用已存在的 GUI 图形窗口。如果 GUI 已存在，MATLAB 将该 GUI 带到前台而不是重新创建一个新的窗口。

如果允许 MATLAB 显示 GUI 的多个实例，则每一个 GUI 调用命令都将创建一个新的窗口。GUIDE 通过在应用程序 M 文件中生成使用 openfig 命令的代码来实现这个特征。使用 reuse 或 new 字符串指定 GUI 的一个或多个实例：

fig = openfig(mfilename,'reuse');

或

fig = openfig(mfilename,'new');

注意确保 openfig 命令在应用程序 M 文件中（包括注释行）仅出现一次。

- 使用系统颜色设置作为背景颜色（Use system colors scheme for background）：GUI 组件使用的颜色与计算机系统有关。选项使图形窗口的背景颜色与用户控件缺省背景颜色（与系统有关）相匹配。图 8-16 说明了选择与不选择该选项之间的效果差异。

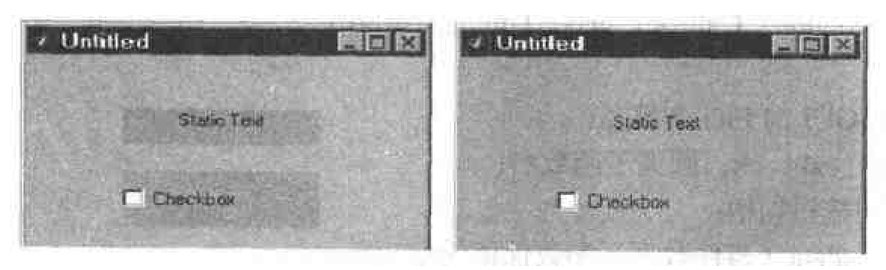


图 8-16　背景颜色使用系统颜色设置与否的比较

GUIDE 在 M 文件中使用以下语句将窗口颜色设置为与组件相匹配的颜色：

set(fig,'Color',get(0,'DefaultUicontrolBackgroundColor'));

注意要确保该语句仅在 M 文件中（包括注释行）出现过一次。

- 当应用程序窗口销毁时函数才返回（Function does not return until application window dismissed）：GUIDE 应用程序选项将生成一个用来等候用户输入的应用程序 M 文件，该文件通过调用 uiwait 函数来阻止 M 文件的继续执行。在执行等待期间 MATLAB 将处理事件序列，也就是说在此期间任何用户与 GUI 的交互都能够调用相应的回调子函数，但是除非以下两个事件发生：GUI 窗口被删除；GUI 中对象的回调函数执行一个 uiresume 命令，否则执行流程总是会返回到 M 文件中。

这个特征使用户能够阻止 MATLAB 命令行的运行直至用户响应该对话框为止，该特征允许同时执行回调函数。将这一选项与模态对话框联合使用将限制用户与对话框的交互。GUIDE 通过以下应用程序 M 文件代码实现这一特征：

uiwait(fig);

其中 fig 是 GUI 图形窗口的句柄。注意确保该语句在文件中只出现一次。

8.3.2　GUI 界面设计

用户可以使用 openfig、open 或 hgload 命令来显示一个 GUI 图形窗口，这些命令将相应的 FIG 文件装载到 MATLAB 的工作平台中，然后用户就可以对该 GUI 进行重新设计。用户也可以使用界面设计编辑器 FIle 菜单的 Open 选项或以下命令来装载一个已存在的 GUI 进行编辑：

guide mygui.fig

GUI 界面设计是通过使用界面设计编辑器进行的。基本的编辑方法在第二节中已经作了详细介绍，这里不再赘述。

GUI 的创建过程经常会要求用户定义组件的 Tag 属性值和 callback 子函数名。GUIDE 给用户 GUI 界面中的每个组件指定一个 Tag 属性值（例如 pushbutton1），然后使用该字符串来命名回调函数（例如 pushbutton1_Callback）。在第一次激活或保存 GUI 之前最好为组件选择一个 Tag 名和文件名。

使用界面设计编辑器 File 菜单的 Save as 选项可以给应用程序 M 文件重命名并重新设置

callbace 属性以保证回调函数的正常运行。注意由于 GUIDE 使用 Tag 属性来命名函数和结构体的域名，所以用户选择的 Tag 必须为一个有效的 MATLAB 变量名。可以使用 isvarname 函数来确定用户指定的字符串是否有效。如果用户在 GUIDE 创建 M 文件和 FIG 文件后对 GUI 进行了修改，那么必须确保用户的代码能够兼容这些修改。

DUIDE 自动给每一个用户控件的 Tag 属性分配一个字符串并使用该字符串进行这样两项工作：构造生成的回调子函数名；给句柄结构体添加一个域名。

如果用户在 DUIDE 生成回调子函数后对 Tag 属性进行了修改，GUIDE 将不会生成一个新的子函数。然而，由于句柄结构体是实时创建的，所以 GUIDE 将在句柄结构体中使用新的 Tag 名。

改变 Tag 将导致用户引用对象句柄时出现一些问题，例如，以下语句：

file_list = get(handles.listbox1,'String');

从一个 Tag 为 listbox1 的列表框中获取 String 属性值，如果将该列表框的 Tag 修改为 file_listbox，那么以后的 GUI 实例需要用户将该语句变为：

file_list = get(handles.file_listbox,'String');

避免这个问题最好的方法就是在添加组件的同时设置 Tag 属性值。如果一定要在应用程序 M 文件创建后修改 Tag 并希望重新命名回调子函数来保持 GUIDE 所使用的名称一致，那么用户必须要做到这样两点：修改句柄结构体所有输出数据参考；修改回调子函数名称。当用户保存或激活 GUI 时，GUIDE 将使用一个在应用程序 M 文件中执行回调子函数的字符串来替换每一个取值为 automatic 的 Callback 属性。当第一次将一个按钮插入界面中时，其 Callback 属性将如图 8-17 所示。

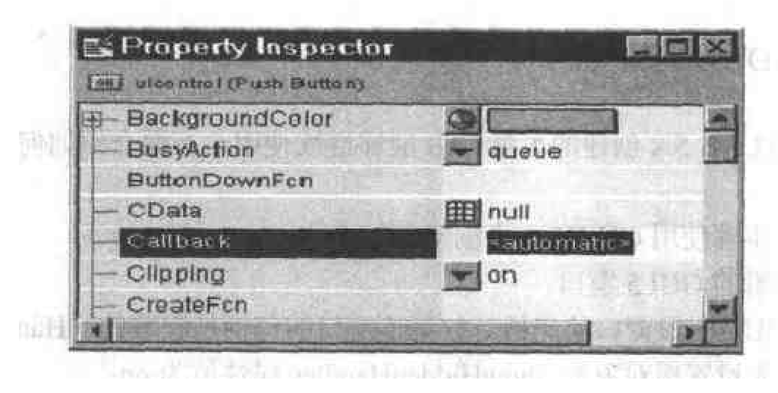


图 8-17　按钮组件的 Callback 属性

用户保存或激活图形窗口时，GUIDE 将修改 Callback 属性为执行回调子函数字符串。图 8-18 表明了添加第六个按钮的字符串。

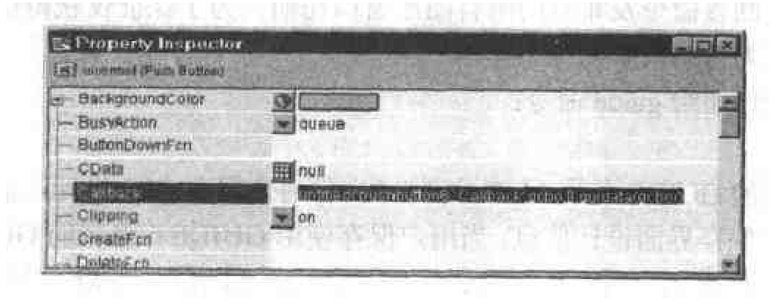


图 8-18　添加第六个按钮的字符串

如果用户希望修改回调子函数的名称，那么必须同时修改用户控件的 Callback 属性字符串。图 8-19 为调子函数重命名为 Closebutton_Callback 后的外观。

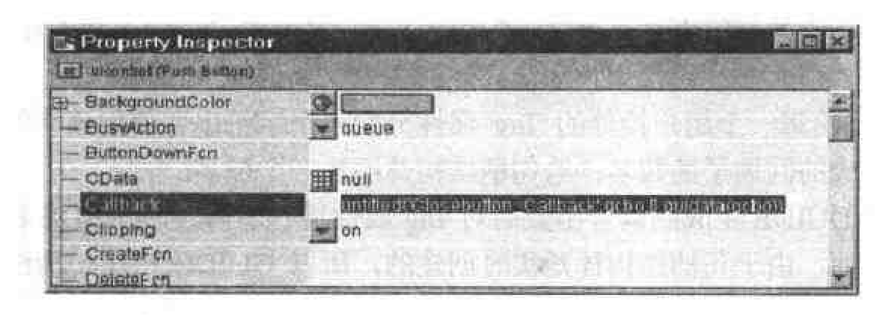


图 8-19　回调子函数重命名后的外观

GUIDE 为其他 callback 属性生成相似的字符串。

GUI 的 FIG 文件和相关的应用程序 M 文件的文件名相同，只是文件扩展名不同。当用户执行 M 文件以发布 GUI 时，使用 mfilename 命令从 M 文件名来确定 FIG 文件名：

```
fig = openfig(mfilename,'reuse');
```

如果 FIG 文件名与 M 文件名不同，以上语句将不会被成功调用。

对于 GUI 实现的第三个步骤——回调函数编程将在下一章中向读者详细介绍。

完成 GUI 的创建工作后，使用 File 菜单下的 Save 或 Save as 选项将用户界面保存为一个 FIG 文件。当用户保存或激活图形窗口时，GUIDE 将自动创建应用程序 M 文件。执行 M 文件来显示 GUI。对于那些用户尚未实现，而 GUIDE 在 M 文件中保存原型的回调函数，运行时 MATLAB 将返回一个消息说明这些函数尚未实现。

8.3.3　使用 GUIDE 6 编辑 GUI 5

为了保证 MATLAB 5.x 创建的大量 GUI 能够继续使用，下面介绍如何使用 GUIDE 6 来编辑 GUI 5。

按照以下三个步骤使用 GUIDE 6 来编辑 GUI 5：

（1）显示待编辑的 GUI 5 窗口；

（2）获取该 GUI 图形窗口的句柄。如果该窗口的句柄是隐藏的（HandleVisibility 属性值为 off），那么首先设置根对象的 ShowHiddenHandles 属性值为 on：

```
set(0,'ShowHiddenHandles','on')
```

然后获取根对象 Chileren 属性值：

```
h = get(0,'Children');
```

以上语句将返回该命令发布时的所有图形窗口句柄。为了保证仅获得待编辑句柄，确保所显示的图形窗口为待编辑 GUI 窗口。

（3）将该句柄传递给 guide 命令：

```
guide(h)
```

MATLAB 5.x 的 GUIDE 使用 MAT 文件/M 文件对来保存 GUI 设计，而在 MATLAB 6.0 中使用 FIG 文件来保存界面设计信息。当用户保存使用 GUIDE 6 编辑的 GUI 5 时，MATLAB 将创建一个 FIG 文件来包含所有界面设计信息，最初的 MAT 文件/M 文件组合将不再使用。为了显示修改后的 GUI，使用 open 或 hgload 命令来装载创建的 FIG 文件或者运行应用程序

M 文件。

当用户保存 FIG 文件时，确保用户 GUI 中用户控件的回调函数属性已经被设置为所需的回调函数字符串或回调函数 M 文件名。如果用户的 GUI 5 使用了一个既包含界面设计代码又包含回调函数子程序的 M 文件，那么用户需要重新构造一个仅包含初始化 GUI 命令和回调函数的 M 文件。缺省情况下，GUIDE 生成的 M 文件与界面设计编辑器保存的 FIG 文件同名，当用户在界面设计编辑器中激活一个 GUI 时，GUIDE 将试图执行相应的 M 文件来发布该 GUI。

8.4　实 例 讲 解

【实例】：实现一个用来显示名字和电话号码的 GUI，该 GUI 中输入的名字和电话号码都保存在一个 MAT 文件中。要求该 GUI 可以添加新条目并将该条目保存在同一个 MAT 文件或一个新的 MAT 文件中。

分析：这个问题要用到以下几种GUI编程技术：

- 使用打开和保存对话框的方法为使用者提供寻址和打开地址簿 MAT 文件、保存 MAT 文件修改、创建新地址簿的方法；
- 定义 GUI 菜单回调函数；
- 使用 GUI 句柄结构体保存和传递全局数据（名字和电话号码）；
- 使用 GUI 图形窗口重画函数实现重新显示工作。

这里仅仅介绍GUI组态和界面设计工作，关于回调函数的编程方法将在下一章的实战讲解中说明。

解决方法：

步骤一：GUI组态。打开GUI应用程序选项对话框，进行如下设置：

- 重画行为：User-specified；
- 命令行可访问性：Off；
- 同时生成 FIG 文件和 M 文件以及：

  生成回调函数原型：Generate callback function prototypes；

  同时只允许一个实例运行：Application allows only one instance to run。

步骤二：进行界面设计。在命令行中键入：guide

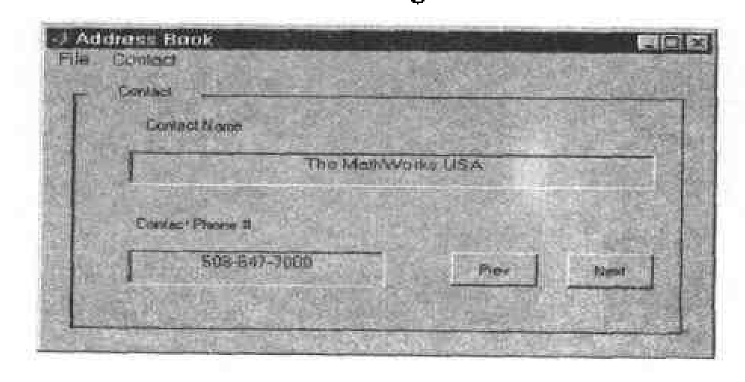


图 8-20　地址簿 GUI 外观

可以按照用户的喜好任意设计 GUI 的界面并排列各组件，例如图 8-20 所示的 GUI 界面。使用 Prev 和 Next 按钮在地址簿中的条目间向前或向后翻页。

步骤三：设计菜单。地址簿GUI包含一个File菜单，其菜单项为Open，GUI用该菜单项来装载地址簿MAT文件。当用户对地址簿进行修改时，用户需要保存被修改的MAT文件或将这些修改保存为一个新的MAT文件，使用File菜单的Save和Save as菜单项使用户能够实现这些要求。

步骤四：对各个组件的属性进行设置。在界面设计编辑器中使用菜单命令打开属性检查器，根据用户的界面需要设置组件的Tag、callback等属性。必须设置Prev和Next按钮的Callback属性以及Save和Save as菜单项的回调函数，使他们分别能够调用同一个回调函数Save_Callback、Prev_Next_Callback。另外，将窗口的ResizeFcn属性设置为：

address_book('ResizeFcn',gcbo,[],guidata(gcbo))

步骤五：保存GUI。激活GUI界面，确保界面符合用户要求。设计满意后保存GUI。

步骤六：执行GUI。运行地址簿GUI的应用程序M文件。可以对该M文件进行反复调试使之符合用户的要求。

8.5 小　　结

在本章中介绍了 GUI 的一些基本创建方法，读者将学会如何使用 GUIDE 的编程工具集来实现自身的 GUI。但是读者对于 GUI 的核心内容——控制 GUI 行为的回调函数编程方法还没有做深入的研究，下一章中将主要介绍回调函数的编程方法。

第9章 深入GUI编程

在上一章的基础上，本章主要介绍GUI的深入编程方法。首先将说明应用程序M文件系统生成代码的具体含义以及利用句柄结构体管理GUI数据的方法，然后介绍GUI组件回调函数的类型和回调函数的中断方法，最后说明如何控制GUI图形窗口的行为。

9.1 M文件以及GUI数据管理

9.1.1 应用程序M文件理解

GUI包含许多可以使软件与用户终端进行交互的用户界面组件，GUI的实现任务之一就是控制这些组件如何响应用户的行为。对应用程序M文件代码进行详细分析的目的就是要通过了解GUIDE创建应用程序M文件的功能，从而实现GUI的规划。

MATLAB通过创建应用程序M文件为GUI控制程序提供一个框架。这个框架孕育着一种高效而坚固的编程方法：所有代码（包括回调函数）都包含在应用程序M文件中，这就使得M文件仅有一个入口可以初始化GUI或调用相应的回调函数以及GUI中希望使用的任意帮助子程序。无论用户是否使用GUIDE来创建应用程序M文件，这里所说的编程技术对用户进行GUI编程都是有用的。

1.回调函数自动命名

GUIDE给添加到应用程序M文件中的回调子函数自动命名。GUIDE还将Callback属性值设置为一个字符串使用户激活控件时该子函数能够被调用。

首先说明GUIDE如何为回调子函数命名。当用户在GUI界面中添加一个组件时，GUIDE为该组件的Tag属性指定一个用来生成回调函数名称的值。例如，假设用户添加到界面中的第一个按钮被称为pushbutton1，当用户保存或激活图形窗口时，GUIDE在应用程序M文件中添加一个名为pushbutton1_Callback的回调子函数。如果用户为该按钮定义了ButtonDownFcn属性，则相应的回调子函数的名称为pushbutton1_ButtonDownFcn。

GUIDE通过指定回调字符串为回调子函数命名。当用户第一次为GUI界面添加一个组件时，其Callback属性将被设置为字符串<automatic>，当用户保存或激活GUI时，该字符串将通知GUIDE使用应用程序M文件中相应的回调子函数名来替换该字符串。例如，GUIDE将设置pushbutton1用户控件的Callback属性为：

```
my_gui('pushbutton1_Callback',gcbo,[],guidata(gcbo))
```

其中，my_gui为应用程序M文件名；pushbutton1_Callback为定义的回调子函数名；gcbo是一个返回回调对象名称的命令；guidata(gcbo)返回句柄结构体。

2. 应用程序 M 文件的执行路径

应用程序 M 文件根据 GUI 调用文件时所传递的参数类型来决定有待执行的行为。例如，如果不向 M 文件传递任何参数，则调用 M 文件将会发布该 GUI（如果此时用户指定了一个 M 文件的输出参数，那么 M 文件将返回 GUI 图形窗口的句柄）；如果使用一个子函数名作为传递给 M 文件的第一个参数，那么调用 M 文件将会执行指定的子函数（通常是回调子函数）。

应用程序 M 文件包含一个调度函数，该函数使 GUI 能够根据调用方式决定执行路径。下面来分析调度函数代码。

应用程序 M 文件调度函数的功能是通过在 if 语句中使用 feval 函数实现的。在调用 M 文件时，feval 函数将执行字符串参数所指定的子函数。feval 函数在一个 try/catch 调试语句块中执行，这是因为当 GUI 试图调用不存在的子函数（找不到与传递参数名称相同的子函数）、调用发生错误时能够得到正确的处理。以下是 GUIDE 生成的调度函数功能代码（用户不能修改）：

```
% 若无输入参数，则打开 GUI
if nargin == 0
    fig = openfig(mfilename,'reuse');
        ...
%如果输入参数为字符串，执行相应子程序
elseif ischar(varargin{1})
    try
        [varargout{1:nargout}] = feval(varargin{:});
    catch
    disp(lasterr);
    end
end
```

任何由子函数返回的输出参数都将通过该函数返回 GUI 中。图 9-1 说明了应用程序 M 文件的执行路径。

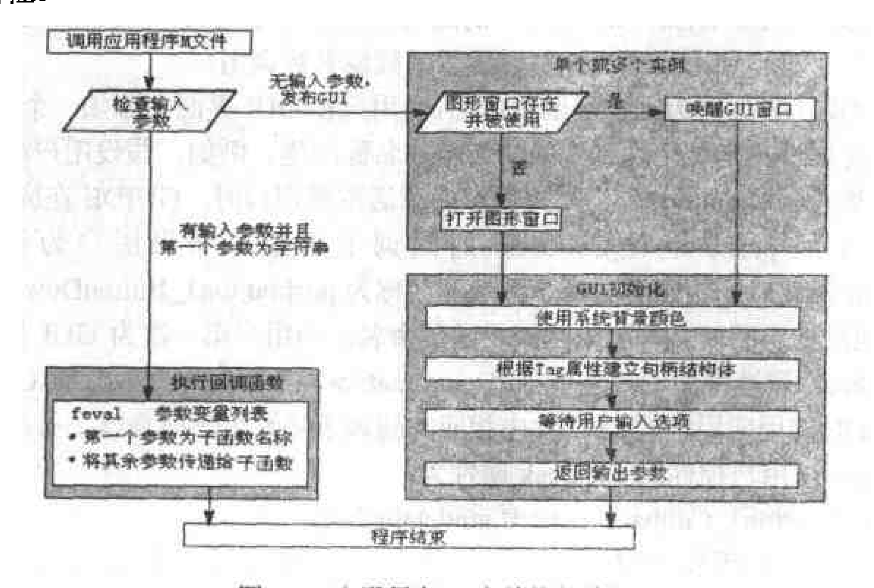


图 9-1　应用程序 M 文件执行路径

虽然 GUIDE 生成的回调子函数的参数是明确的，但是参数列表的长度是变化的，这是由于输入参数 varargin 其实可以是多个参数，用户可以在调用 M 文件时通过该参数给被调用的子函数赋予任意多个用户所需的参数。为了传递额外的参数，通过编辑 Callback 属性字符串来包括这些参数。例如，应用程序 M 文件自动生成的回调函数代码如下：

```
my_gui('pushbutton1_Callback',gcbo,[],guidata(gcbo))
```

使用属性检查器对该字符串进行修改，使之包括额外的参数：

```
my_gui('pushbutton1_Callback',gcbo,[],guidata(gcbo),arg1,arg2)
```

此时子函数 pushbutton1_Callback 就可以包含多个不同的输入参数，调用格式如下：

```
varargout = pushbutton1_Callback(h,eventdata,handles,varargin)
```

3. GUI 初始化

首先，应用程序 M 文件使用 openfig 命令来装载 GUI 图形窗口，格式如下：

```
fig = openfig(mfilename,'reuse');
```

一定要注意这个语句打开的 FIG 文件名是源于应用程序 M 文件的（mfilename 命令返回当前执行的 M 文件名），如果用户使用由 GUIDE 创建的应用程序 M 文件，那么用户必须保证 FIG 文件与 M 文件同名。参数 reuse 决定任何时候都只能有一个 GUI 实例在运行。

应用程序 M 文件自动包含一些管理 GUI 的有效技术，这些技术包括：

- 单个/多个实例控制：当设计 GUI 时，用户必须明确选择是否允许 GUI 图形窗口的多个实例同时存在。如果选择不允许，也就是说同一时刻只有一个实例存在，那么以后试图创建另一个 GUI 的操作将仅仅导致已存在的 GUI 出现在其他窗口之上。很多信息对话框（尤其是模态对话框）只能同时存在一个实例，因为对于用户的某种特定行为，对话框只能提示一次。GUIDE 界面设计编辑器实际上是一个允许多个实例存在的 GUI，这样设计这个 GUI 的目的是使用户能够同时打开多个界面；
- GUI 图形窗口在屏幕中的位置不受计算机屏幕和分辨率的影响：应用程序 M 文件使用 movegui 命令来确保 GUI 图形在目标计算机的屏幕上可见，使 GUI 不受计算机屏幕的尺寸和分辨率的影响。如果指定的窗口位置将导致 GUI 窗口位于屏幕之外，那么 movegui 命令将窗口移动到屏幕中离指定位置最近的地方。movegui 语句格式如下：movegui(fig,'onscreen')。其中 fig 是由 openfig 命令返回的 GUI 图形窗口句柄。
- 自动创建 GUI 组件句柄结构体；
- 自动命名 Tag 属性、生成子函数原型并指定回调属性字符串：当用户发布 GUI 时，应用程序 M 文件创建一个包含所有 GUI 组件句柄的结构体，然后将该结构体保存在图形窗口的应用程序数据中以备将来的使用。句柄结构体的域名与相应对象的 Tag 属性值一致，例如，一个 Tag 属性值为 pushbutton1 的对象可以使用以下域名访问：handles.pushbutton1

用户可以使用相同的方法访问隐藏的图形窗口句柄，例如假设窗口的 Tag 属性为 figure1，则 handles.figure1 就是图形窗口的句柄。

应用程序 M 文件使用 guihandles 和 guidata 来创建并存储句柄结构体：

```
handles = guihandles(fig);
guidata(fig,handles);
```

注意只有那些 Tag 属性值为有效字符串的对象的句柄才能够保存在句柄结构体中。可

以使用 isvarname 来确定字符串是否有效。

句柄结构体是传递给所有回调函数的参数之一，因而用户可以使用这个结构体来保存数据并在子函数间传递。

● 单个 M 文件同时包含 GUI 初始化和回调函数执行代码。

9.1.2　GUI 数据管理

GUIDE 使用保存在 GUI 图形窗口中的应用程序定义数据实现数据的存储和获取机制。GUIDE 使用这个机制来存储一个包含所有 GUI 组件对象句柄的结构体。由于这个结构体将传递给每一个对象的回调子函数，所以该结构体也可以用来保存其他数据。下面的例子说明了如何使用句柄结构体在回调子函数间传递数据。

假设用户希望创建一个包含滚动条和编辑框的 GUI 并实现以下行为：

● 当用户移动滚动条时编辑框将显示新的数值；

● 当用户在编辑框中键入新的数值时，滚动条将会刷新为新的数值；

● 如果编辑框输入数据超出范围，应用程序将返回一个错误信息说明输入数据中出现了多少个错误。

图 9-2　示例 GUI 外观

图 9-2 显示了该 GUI 的外观。

实现该 GUI 可以分为以下几个步骤：

步骤一：在初始化时定义数据域。

以下的 GUI 创建代码使用了两个句柄结构体域：errorString 和 numberOfErrors。函数 guihandles 创建结构体并将滚动条和编辑框的句柄添加到句柄结构体相应域中（Tag 属性值作为域名），函数 guidata 将结构体存储在图像窗口应用程序数据中：

```
fig = openfig(mfilename,'reuse');
handles = guihandles(fig);
handles.errorString = 'Total number of errors: ';
handles.numberOfErrors = 0;
guidata(fig,handles);
```

步骤二：在滚动条对象的回调函数中设置编辑框的数值。

使用句柄结构体来获得编辑框和滚动条对象的句柄，然后按照滚动条的 Value 属性设置编辑框的 String 属性：

```
set(handles.edit1,'String',...
num2str(get(handles.slider1,'Value')));
```

注意 GUIDE 生成的子函数是以句柄结构体作为输入参数的，因此没有必要通过调用 guidata 函数来获得句柄结构体。然而，如果用户需要对结构体进行修改，那么必须使用 guidata 来保存这些修改。

步骤三：在编辑框的回调函数中设置滚动条的数值。

编辑框回调子函数首先将查询用户输入是否为一个单独的、在滚动条范围内的数值，如果是，则按照这个输入来设置滚动条的数值；如果数值超出滚动条的范围，那么错误计数就

会增加，回调函数将错误字符串和错误计数值同时显示出来：

```
val = str2num(get(handles.edit1,'String'));
if isnumeric(val) & length(val)==1 & ...
        val >= get(handles.slider1,'Min') & ...
    val <= get(handles.slider1,'Max')
set(handles.slider1,'Value',val);
else
% 增加错误计数并显示
handles.numberOfErrors = handles.numberOfErrors+1;
set(handles.edit1,'String',...
    [handles.errorString,num2str(handles.numberOfErrors)]);
guidata(gcbo,handles); %保存所作的修改
end
```

步骤四：更新 GUI 数据。

注意一旦修改了句柄结构体的数值就必须使用 guidata 来保存句柄结构体。guidata 函数可以通过任何一个子对象的句柄来自动判断其父窗口，这对于用户 GUI 不支持命令行访问（不能使用 findobj 函数来获得窗口句柄）的情况非常有用。

即使用户自己编写应用程序 M 文件并且不希望使用句柄结构体，用户仍然可以使用 GUI 图形窗口应用程序数据来存储任何用户希望传递给子函数的数据，可以通过以下步骤实现这一目的：

- 创建一个包含用户希望保存数据的结构体；
- 在图形窗口应用程序数据中保存结构体；
- 在需要时在子函数中获取该结构体。

首先使用 guidata 函数创建数据结构体。guidata 函数为图形窗口应用程序数据提供一个方便的接口，它使用户能够在不知道图形窗口句柄的条件下访问数据，避免了在整个用户资源代码中为应用程序数据创建和保存句柄变量。例如，用户可以使用以下代码来实现访问：

- 应用程序 M 文件中 GUI 初始化代码：

```
fig = openfig(mfilename,'new'); % open GUI and save figue handle
    ...
    data.field1 = value1; % create a structure
    guidata(fig,data) % save the structure
```

- 回调子函数中的代码：

```
data = guidata(gcbo); % load the data
data.field1 = new_value; % change the structure
guidata(gcbo,data) % save the structure
```

注意一旦回调子函数开始执行，guidata 就能够使用 gcbo（其回调函数正在被调用的对象的句柄）获得图形窗口的句柄。然而，在初始化过程阶段，由于没有回调函数被调用，所以不能使用 gcbo。这种情况下可以使用 openfig 函数返回的 GUI 图形句柄。

这里有必要对应用程序定义数据进行说明。应用程序定义的数据提供了应用程序使用 GUI 保存和获取数据的方法。这项技术使用户能够创建对象必需的属性，用户可以使用这

个属性来存储数据。表 9-1 列出了能够访问应用程序数据定义数据的函数，用户可以使用这些函数对 GUI 程序的数据进行查询、修改等操作。

表 9-1 访问应用程序数据定义数据的函数

函 数 名	函 数 功 能
Setappdata	定义应用程序数据
Getappdata	返回指定名称的应用程序数据
Isappdata	如果应用程序数据存在则为真
Rmappdata	删除指定名称的应用程序数据

9.2 回调函数的使用方法

9.2.1 回调函数类型

实现一个 GUI 的首要机制就是对构成用户界面的用户控件的回调函数进行编程。除了用户控件的 Callback 属性，还可以使用其他一些属性来定义回调函数。

1. 所有图形对象的回调函数属性

所有图形对象都有三个能够定义回调函数的属性：

ButtonDownFcn：当用户将鼠标放置在某个对象或对象相邻的 5 个像素范围内时，如果点击鼠标左键，MATLAB 将会执行这个回调函数；

CreateFcn：MATLAB 在创建对象时调用这个回调函数；

DeleteFcn：MATLAB 在删除对象之前调用这个回调函数。

2. 图形窗口的回调属性

图形窗口有如下所述的几种额外的、用来执行相应用户行为的属性：

CloseRequestFcn：当请求关闭图形窗口时 MATLAB 将执行这个回调函数；

KeyPressFcn：当用户在图形窗口内按下鼠标键时 MATLAB 将执行这个回调函数；

ResizeFcn：当用户重画图形窗口时 MATLAB 将执行这个回调函数；

WindowButtonDownFcn：一旦用户在图形窗口内无控件的地方按下鼠标键时，MATLAB 就会执行这个回调函数；

WindowButtonMotionFcn：当用户在图形窗口中移动鼠标时 MATLAB 将执行这个回调函数；

WindowButtonUpFcn：当用户在图形窗口中释放鼠标键时 MATLAB 将执行这个回调函数。

MATLAB 将根据用户的行为来判断究竟执行哪个回调函数。点击一个有效的用户控件将会阻止任何 ButtonDownFcn 和 WindowButtonDownFcn 回调函数的执行，而如果用户点击一个无效用户控件、图形窗口或其他定义了回调函数的图形对象时，MATLAB 将首先执行图形窗口的 WindowButtonDownFcn 函数，然后再执行鼠标点击对象的 ButtonDownFcn 函数。

3. 各种组件的回调函数

下面将介绍常用组件的回调函数使用方法。

（1）按钮和拴牢按钮：按钮和拴牢按钮的回调函数需要对按钮的状态进行查询以决定其当前的状态。用户可以使用当前回调对象的句柄（gcbo）来实现这项工作：

```
get(gcbo,'Value')
```

当按钮按下时，MATLAB 将 Value 属性值（缺省为 1）设置为 Max 的属性值，没有按下时设置为 Min 的属性值。

另外，可以为按钮或拴牢按钮设置表面图像。将 CData 属性值定义为一个 m×n×3 的 RGB 值矩阵可以给按钮定义一个真彩图像。例如，数组 a 定义了一幅使用随机颜色（取值从 0 到 1）的 16×128 真彩图像：

```
    a(:,:,1) = rand(16,128);
    a(:,:,2) = rand(16,128);
    a(:,:,3) = rand(16,128);
set(h,'CData',a)
```

（2）单选按钮：单选按钮通常以组为单位进行操作，每一个单选按钮都有两种状态：被选中和未被选中。用户可以通过使用按钮的 Value 属性来查询或设置单选按钮的状态。如果 Value 取值为 Max（即为 1），则该按钮被选中，如果为 Min（0）则未被选中。要实现一组单选按钮间的相互排斥，就要设置按钮组中所有按钮的 Value 属性值为 0，由 MATLAB 将被选中按钮的 Value 设置为 1。以下定义的函数可以作为应用程序 M 文件中任意一个单选按钮的回调函数，该函数以按钮组中其他未被选中按钮的句柄作为参数：

```
function mutual_exclude(off)
set(off,'Value',0)
```

单选按钮的句柄可以通过句柄结构体获得。以下是包含四个单选按钮的按钮组中第一个按钮的回调函数 mutual_exclud e 的代码：

```
    function varargout = radiobutton1_Callback(h,eventdata,handles,varargin)
    off = [handles.radiobutton2,handles.radiobutton3,handles.radiobutton4];
mutual_exclude(off)
```

在设置了单选按钮的正确状态后，回调函数可以继续执行指定的操作。

（3）复选按钮：复选按钮的 Value 属性表明了复选按钮当前的状态。如果 Value 的取值为 Max 则表示选中，否则表示未选中。用户可以通过在回调函数中查询对象的 Value 属性确定对象的状态，代码如下：

```
    function checkbox1_Callback(h,eventdata,handles,varargin)
    if (get(h,'Value') == get(h,'Max'))
    then checkbox is checked-take approriate action
    else
    checkbox is not checked-take approriate action
end
```

（4）滚动条：用户可以通过使用滚动条对象的 Position 属性来定义滚动条的宽度和高度，如果宽度大于高度则表示为一个水平滚动条，否则表示为一个竖直滚动条。例如图 9-3 所示

的设置将创建一个水平滚动条。

图 9-3　滚动条对象的属性设置

有四种控制滚动条的范围和滚动尺寸的属性：

Value：包含当前滚动条的位置信息；

Max：定义滚动条取值范围的上限；

Min：定义滚动条取值范围的下限；

SliderStep：指定滚动条相对于整个滚动区域的滚动尺寸。

用户可以通过设置 Value 属性来设置滚动条的初始位置并在滚动条回调函数中查询该属性获得用户的当前设置，例如：

```
slider_value = get(handles.slider1,'Value');
```

SliderStep 属性控制用户在使用滚动条时滚动条对象 Value 属性的变化量。可以使用一个两元素的变量来定义 SliderStep 属性值，缺省情况下该属性值为[0.01 0.10]，表示每点击一次滚动条的箭头按钮，滚动条将移动整个滚动范围的 1%，而点击一次滚动槽将移动整个范围的 10%。真正的滚动尺寸是一个滚动范围和点击次数的函数。

假设用户希望建立一个如下指标的滚动条：滚动范围：5 到 8；箭头滚动尺寸：0.4；滚动槽滚动尺寸：1；初始位置：6.5。

首先设置滚动条对象的相应属性。在用户的应用程序 M 文件初始化部分添加以下代码：

```
    slider_step(1) = 0.4/(8-5);
    slider_step(2) = 1/(8-5);
set(handles.slider1,'sliderstep',slider_step, 'max',8,'min',5,'Value',6.5)
```

当用户释放鼠标键时将会执行滚动条的回调函数。

（5）组合框：组合框没有与自身相联系的回调函数，并且只有控件才能够出现在组合框中（坐标轴不能）。组合框是不透明的，如果需要在组合框中添加组件，那么需要使用 Layout 菜单的 Bring to Front 和 Send to Back 操作来实现组件的可视化；

（6）列表项：列表项的 Value 属性包含被选列表项在字符串列表中的索引号，第一个列表项的索引为 1。如果用户选择多项，Value 就是一个索引列表。缺省情况下，列表框第一次显示时第一个列表项被加亮。如果用户不希望任何一项被加亮，那么将 Value 属性置为空。列表项的 Max 属性值和 Min 属性值决定了用户是否能够选择多个列表项。如果 Max 与 Min 的差值大于 1，那么就允许多个列表项被选中；如果小于等于 1，那么就不允许选择多项。

列表框区分列表项的单击和双击操作，并相应地将 SelectionType 属性设置为 normal 或

open。MATLAB 在列表项被选中、Value 属性值发生变化后调用回调函数（点击列表框的滚动槽将不会调用回调函数）。在这种情况下，用户不需要给列表框添加回调函数，而是需要添加另一个组件（例如确认按钮），并对该组件的回调函数进行编程，从而查询列表框的 Value 属性值以及 SelectType 属性值。如果用户使用的是自动生成的应用程序 M 文件，可以有两种设置方法：将列表框的 Callback 属性设置为空字符串并在应用程序 M 文件中删除回调子函数原型，或者保留回调子函数但是将缺省的 disp 语句删除，使用户选择列表项时不会导致任何代码执行。如果用户确信不会调用列表框的回调函数，并且希望应用程序 M 文件简捷有效，最好使用第一种方法；如果用户可能会需要调用列表框的回调函数，使用第二种方法。

（7）文本菜单：不作选择时文本菜单将显示由 Value 属性指定索引号决定的当前选项，菜单的第一项索引号为 1。用户可以在回调函数中查询 Value 属性确定当前的选项；

（8）菜单：当用户选择菜单项时将执行菜单回调函数。如果回调函数是一个非常简单的命令，那么可以直接在菜单编辑器的文本框内键入要执行的回调函数代码，但最好还是在应用程序 M 文件中添加一个子函数作为菜单的回调函数并将该函数名键入到文本框中。注意 GUIDE 并不是自动给菜单项添加应用程序 M 文件回调子函数的，用户必须手动添加。同时用户必须使用回调函数正确的语法格式，例如，Select All 菜单项必须给出以下回调字符串：

MyGui('menu_Edit_SelectAll_Callback',gcbo,[],guidata(gcbo))

其中：

MyGui：用来发布包含该菜单项图形窗口的应用程序 M 文件名；

menu_Edit_SelectAll_Callback：Select All 菜单项的回调子函数名；

gcbo：Select All 用户菜单项的句柄；

[ ]：保留空矩阵；

guidata(gcbo)：从图形窗口的应用程序数据中获取句柄结构体。

如果确认回调函数字符串是有效的，那么将该字符串键入菜单编辑器的 Callback 文本框中，然后在 MyGui.m 文件中添加子函数 menu_Edit_SelectAll_Callback。

4. *其他常用回调函数使用方法*

（1）控件使能：用户可以通过设置控件的 Enable 属性来控制一个控件是否响应用户的鼠标输入。控件有以下三种状态：

on：控件正在被操作；

off：控件不能响应输入，同时该控件的标签变灰；

inactive：控件不能响应输入但标签不变灰。

当一个控件被设置为不能响应输入时，使用鼠标点击该控件将不会调用相应的回调函数，但是另外两个回调函数将被执行：图形窗口的 WindowButtonDownFcn 回调函数；控件的 ButtonDownFcn 回调函数。如果给无效控件定义了文本菜单，那么右击该对象时菜单仍会出现。

（2）坐标轴：坐标轴使 GUI 可以作为图形输出目标，也就是说使得 GUI 具有绘图和显示图像的功能。坐标轴不是一个用户控件对象，但是可以通过编程使用户界面包含坐标轴以及显示在坐标轴上的图形；

（3）图形窗口：图形窗口是用户使用界面设计编辑器设计的 GUI 的显示窗口。如果不

希望图形窗口作为命令行或其他 GUI 绘图命令的输出目标，可以将 HandleVisibility 和 IntegerHandle 属性设置为 off。但是这同时也意味着图形窗口对于用户自身的 GUI 也是隐藏的。为了在用户 GUI 中发布绘图命令，用户应该在创建图形窗口和坐标轴时保存它们的句柄（可以使用句柄结构体），然后将坐标轴的父对象设置为相应的图形窗口，将由绘图命令生成图形的父对象设置为相应的坐标轴。具体步骤是这样的：创建图形窗口时保存句柄；创建坐标轴，保存句柄，设置其 Parent 属性为图形窗口的句柄；创建图形，保存句柄，设置其 Parent 属性为坐标轴的句柄。

以下代码说明了这几个步骤：

```
fHandle = figure('HandleVisibility','off','IntegerHandle','off',...
'Visible','off');
aHandle = axes('Parent',fHandle);
pHandles = plot(PlotData,'Parent',aHandle);
set(fHandle,'Visible','on')
```

9.2.2 回调函数执行中断

缺省情况下 MATLAB 允许正在执行的回调函数被后来调用的回调函数中断。例如，假设用户创建了一个在装载数据时能够显示一个进展条的对话框，这个对话框包含一个取消按钮可以阻止数据的装载操作，那么取消按钮的回调函数将会中断正在执行的数据装载子函数。某些情况下用户可能不希望正在执行的回调函数被用户的行为中断，例如，在重新显示一幅图形之前可能会需要使用一个数据分析工具进行数据流长度计算。假设用户行为可以中断回调或函数的执行，此时如果用户无意中点击鼠标导致回调函数执行中断，那么就可能导致 MATLAB 在返回原来的回调函数之前发生状态改变，引起执行错误。

1. 可中断设置

所有图形对象都有一个控制其回调函数能否被中断的属性 Interruptible，该属性的缺省值是 on，表示回调函数可中断。然而 MATLAB 只有在遇到一些特定的命令（drawnow、figure、getframe、pause 和 waitfor）时才会执行中断，转而查询事件序列，否则将会继续执行正在执行的回调函数。在回调函数中出现的计算或指定属性值的 MATLAB 命令将会被立即执行，而影响图形窗口状态的命令或行为将产生被放置在事件序列中的事件。事件可以由被任何导致图形窗口重画的命令或用户行为引发，例如定义了回调函数的鼠标移动行为。仅仅当回调函数执行完毕或回调函数包含命令 drawnow、figure、getframe、pause 和 waitfor 时，MATLAB 才进行事件序列的处理。

如果在回调函数的执行过程中遇到上述某个命令时，MATLAB 将现在执行的程序挂起，然后处理事件序列中的事件。MATLAB 控制时间的方式依赖于事件类型和回调函数对 Interruptible 属性的设置：只有在当前回调对象的 Interruptible 属性值为 on 的情况下，导致其他回调函数执行的事件才可以真正执行要执行的回调函数；导致图形窗口重画的事件将无视于回调对象的 Interruptible 属性值而无条件地执行重画任务；对象的 DeleteFcn 属性和 CreateFcn 属性或图形窗口的 CloseRequestFcn 属性或 ResizeFcn 属性定义的回调函数将无视于对象的 Interruptible 属性而中断正在执行的回调函数。

所有对象都具有一个 BusyAction 属性，该属性决定不允许中断的回调函数执行期间发生的事件将如何处理。BusyAction 有两种可能的取值：queue：将事件保存在事件序列中并等待不可中断回调函数执行完毕后处理；cancel：放弃该事件并将其从事件序列中删除。

2. 回调函数执行期间的事件处理

以下几种情况描述了 MATLAB 在一个回调函数的执行期间是怎样处理事件的：

- 如果遇到了 drawnow、figure、getframe、pause、waitfor 命令之一，那么 MATLAB 将该回调函数挂起并开始处理事件序列；
- 如果事件序列的顶端事件要求图形窗口重画，MATLAB 将执行重画并继续处理事件序列中的下一个事件；
- 如果事件序列的顶端事件将会导致一个回调函数的执行，MATLAB 将判断回调函数被挂起的对象是否可中断：如果回调函数可中断，MATLAB 执行与中断事件相关的回调函数；如果该回调函数包含 drawnow、figure、getframe、pause、waitfor 命令之一，那么 MATLAB 将重复以上步骤；如果回调函数不可中断，MATLAB 将检查事件生成对象的 BusyAction 属性；如果该属性值为 queue，Maltab 将事件保留在事件序列中；如果为 cancel 则放弃该事件；
- 当所有事件都被处理后，MATLAB 恢复被中断函数的执行。

这些步骤将一直持续到回调函数执行完毕为止。当 MATLAB 返回命令窗口时，所有残余的事件都已经被处理了。当然，由于序列中事件的类型不同，以上步骤有可能不一一执行。

9.3 GUI 图形窗口控制

9.3.1 GUI 图形窗口行为控制

在设计 GUI 时，需要考虑显示 GUI 时图形窗口将怎样展开。一个 GUI 图形窗口的行为是否恰当地依赖于其使用目的。考虑以下几种情况：

- 一个实现图形注释的工具 GUI 通常设计成其他 MATLAB 执行任务可访问的 GUI，这个工具可能一次只能对一幅图形进行注释，所以每一幅图形都需要一个新的工具实例；
- 一个对话框，能向用户发出询问并阻止 MATLAB 运行直至用户作出回答。但是用户可能需要通过观察其他的 MATLAB 窗口获得信息后才能够对该对话框作出回答；
- 一个警告用户其指定的操作将会破坏文件的对话框，该对话框能够在执行用户所需的操作前强迫用户作出回答。此时的图形窗口既被阻止又是模态的（用户不可观察其他窗口）。

以下三种技术能够有效地实现以上 GUI 的设计要求：

- 允许单个或多个 GUI 实例同时运行；
- 在显示 GUI 时阻止 MATLAB 的运行；
- 使用模态图形窗口使用户只能与当前执行的 GUI 进行交互。

模态图形窗口能够捕捉在 MATLAB 窗口任何可见位置处发生的键盘和鼠标事件。这就意味着一个模态图形窗口能够处理用户与任何组件的交互操作，但是不允许用户访问其他的 MATLAB 窗口（包括命令窗口）。另外，除非删除该模态窗口，否则窗口将始终位于窗口堆栈的最上方。如果用户希望在进行 MATLAB 其他操作之前必须先响应 GUI，那么最好使用模态图形窗口。

下面说明如何使一个 GUI 窗口模态化。事实上设置图形窗口的 WindowStyle 属性为 modal 就可将图形窗口模态化。用户可以使用属性检查器，也可以通过在应用程序 M 文件的初始化阶段调用以下语句进行属性设置：

```
set(fig,'WindowStyle','modal')
```

在这个语句中，set 函数使用由 openfig 函数返回的句柄实现设置工作。

如果要释放模态窗口的控制权，必须在模态窗口 GUI 的回调函数中使用以下方法之一：

- 删除窗口：delete(figure_handle)
- 使窗口不可见：set(figure_handle,'Visible','off')
- 修改窗口的 WindowStyle 属性值为 normal：set(figure_handle,'WindowStyle','normal')

用户还可以在一个模态窗口中使用 Ctrl+C 将该窗口转换为普通窗口。

无论使用哪一种方法释放模态窗口控制权都需要获得该窗口的句柄。由于大多数的 GUI 将图形窗口句柄隐藏以避免无意识的访问，所以在回调函数中使用 gcbf 命令能够最为有效地获取窗口句柄。gcbf 返回当前回调对象所在窗口的句柄，这个句柄就是释放控制权所需的句柄。例如，假设用户对话框包括一个用来关闭对话框的按钮，其回调函数末尾处将调用 delete 来删除对话框：

```
function varargout = pushbutton1_Callback(h,eventdata,handles,varargin)
%执行程序代码
...
%用户响应对话框后删除窗口
delete(gcbf)
```

9.3.2. 设计平台兼容性

用户可以使用以下属性设置来创建一个 GUI，使该 GUI 在不同操作系统下运行时能够保持界面一致：

- 使用缺省字体（用户控件 FontName 属性）；
- 使用缺省的背景颜色（用户控件 BackgroundColor 属性）；
- 使用图形窗口像素单位（图形窗口的 Units 属性）。

1. 使用系统字体

缺省情况下用户控件使用操作平台的缺省字体。例如，如果在 PC 上显示 GUI，那么 GUI 将使用 MS San Serif 字体。如果用户将 FontName 属性设置为 default，则 MATLAB 会确保 GUI 运行时使用系统的缺省字体。当用户 GUI 使用缺省字体在不同操作系统下运行时，用户 GUI 与其他应用程序的 GUI 外观是一致的。

如果用户希望用户控件使用固定宽度的字体，那么可以将 FontName 属性设置为

fixedwidth，这将确保用户 GUI 使用操作系统的标准固定宽度字体。用户可以通过查询 root 对象的 FixedWidthFontName 属性来确定操作系统使用的固定宽度字体的名称，查询语句结构如下：

get(0,'FixedWidthFontName')

用户也可以给 FontName 指定一个具体的字体名称，然而这样做将会对用户 GUI 在不同计算机下运行时的外观产生影响。如果目标计算机没有这种指定的字体，那么操作系统将会用另外一种外观效果可能较差或非标准的字体来显示用户的 GUI，更何况相同名称的字体还有版本之分，不同版本对一个给定字符集的大小要求可能不同。

2. 使用标准背景颜色

缺省情况下用户控件在 GUI 运行时使用操作系统标准的背景颜色，用户 GUI 在不同的操作系统下展开将使用计算机相应的标准颜色。将 BackgroundColor 属性设置为 default 即可采用标准的背景颜色，这种设置将使得用户 GUI 在不同操作系统下保持外观一致。如果用户将 BackgroundColor 属性设置为其他值，MATLAB 将使用用户指定的颜色。

3.使用图形窗口像素单位

具有平台兼容性的 GUI 必须在不同大小和分辨率的计算机上具有相同的外观。由于在不同的计算机上像素的大小可能不同，所以使用图形窗口缺省的像素 Uints 也不能保证 GUI 能够在所有的操作系统下外观一致。

图形窗口的字符单位是由使用系统默认字体的字符确定的：图形窗口字符的单位宽度是字母 x 的宽度，字符的单位高度是文本基线间的距离。设置图形窗口的 Units 属性为 characters 可以使 GUI 在不同计算机上显示时能够自动调整组件间的距离和组件大小。例如，如果一个组件文本标签由于系统字体矩阵的不同而变大，那么组件及组件间的距离也相应按比例地增大。缺省情况下 GUIDE 将图形窗口的 Units 属性设置为 characters。

9.4 实例讲解

【实例】：实现上一章实例讲解中地址簿 GUI 的行为控制。

分析：该GUI中最重要的技术之一就是保证信息跟踪正确并使得不同的子函数能够获得该信息。这里所指的信息包括这样几个部分：当前MAT文件名；MAT文件中的名字和电话号码；一个表明当前姓名和电话号码的索引指针（用户在地址簿中翻页时该指针必须刷新）；图形窗口位置和大小；所有GUI组件的句柄。为了能够正确地管理全局数据，地址簿GUI的行为控制主要实现以下几个方面：

- 发布 GUI：由于 GUI 需要在其他 MATLAB 任务运行的同时执行，所以 GUI 需要设计成不可中断、非模态的；
- 将一个地址簿装入阅读器中：有两种方式可以将地址簿 MAT 文件装入 GUI 中：发布 GUI 时使用参数指定 MAT 文件，如果用户不使用参数，则 MATLAB 装入缺省的地址簿文件 addrbook.mat；用户使用 File 菜单的 Open 选项来装载其他 MAT 文件；

- 管理菜单回调行为：选择 Open 菜单项后将显示一个对话框（uigetfile）使用户能够浏览文件。对话框将返回文件名及其路径，并将返回值传递给 fullfile，从而确保路径对于任意的操作系统而言是合理构建的；Save 和 Save as 菜单用来保存对 MAT 文件的修改；Create New 菜单将简单地清除 Contact Name 和 Contact Phone#编辑框的内容以便于添加新的名字和电话号码。在添加新条目后，用户使用 Save 或 Save as 菜单项来保存地址簿；
- 管理按钮组件回调行为：Contact Name 编辑框显示地址簿条目的姓名，如果用户键入新的姓名，若当前地址簿中已存在该姓名，相应的电话号码将被显示；如果该姓名不存在，则显示一个询问对话框(questdlg)使用户确认是否创建一个新的条目，如果不创建则返回到先前显示的姓名；Contact Phone #编辑框显示与 Contact Name 编辑框条目相匹配的电话号码。如果键入一个新的数字并回车，回调函数将发布一个询问对话框使用户确认是否要对已存在的号码进行修改。按钮控件 Prev 和 Next 用来实现地址簿条目间的翻页；
- 地址簿将自定义重画函数，允许用户增大窗口的宽度以适应长姓名和电话号码，但是不允许缩小窗口宽度或改变窗口高度（这一限制不会限制 GUI 的使用，并且简化了必须保持窗口尺寸与组件大小相匹配的重画函数代码）。用户重画窗口并释放鼠标后将执行重画函数，重画后的窗口尺寸将被保存下来。

解决方法：

步骤一：发布GUI。用户可以不使用参数来调用应用程序M文件，在这种情况下GUI使用缺省的地址簿MAT文件。用户也可以在参数中指定一个MAT文件，这就需要修改缺省的应用程序M文件GUI初始化代码。修改后的初始化代码如下：

```
function varargout = address_book(varargin)
if nargin <= 1
% GUI发布
fig = openfig(mfilename,'reuse');
set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
handles = guihandles(fig);
guidata(fig, handles);
if nargin == 0
    %装入缺省的地址簿
    Check_And_Load([],handles)
elseif exist(varargin{1},'file')
    Check_And_Load(varargin{1},handles)
else
    %如果文件不存在，则返回一个错误对话框并将文本设置为空字符串
    errordlg('File Not Found','File Load Error')
    set(handles.Contact_Name,'String','')
    set(handles.Contact_Phone,'String','')
end
if nargout > 0
```

```
        varargout{1} = fig;
    end
    elseif ischar(varargin{1})
        %调用指定的子函数或回调函数
        try
            [varargout{1:nargout}] = feval(varargin{:});
        catch
        disp(lasterr);
    end
end
```

步骤二：将一个地址簿装入阅读器中。首先要确认MAT文件。作为一个有效的地址簿文件，MAT文件必须包含一个称作Addresses的结构体，该结构体有两个域，分别称为Name和Phone。Check_And_Load子函数将按照以下步骤来确认并装载这些数据：装载指定的或缺省的文件；判断MAT文件是否为一个有效的地址簿文件，如果有效则显示数据，否则显示错误对话框；对于有效的MAT文件，返回1，否则返回0（该返回值由Open菜单项回调函数使用）；在句柄结构体中保存条目MAT文件名和Addresses结构体；指示当前所显示的名字和地址的索引指针。

Check_And_Load代码如下：

```
function pass = Check_And_Load(file,handles)
% 初始化变量pass以判断文件是否有效.
pass = 0;
%如果不指定文件名则使用缺省名称，否则如果指定文件存在就装载它
if isempty(file)
file = 'addrbook.mat';
handles.LastFile = file;
guidata(handles.Address_Book,handles)
end
if exist(file) == 2
data = load(file);
end
% 判断MAT文件是否有效，当存在一个名为Addresses的变量
% 并且其两个域名为Name和Phone时，该文件有效
flds = fieldnames(data);
if (length(flds) == 1) & (strcmp(flds{1},'Addresses'))
fields = fieldnames(data.Addresses);
if (length(fields) == 2) &(strcmp(fields{1},'Name')) &
        (strcmp(fields{2},'Phone'))
    pass = 1;
end
end
```

```
% 如果文件有效则显示
if pass
% 给句柄结构体添加地址
handles.Addresses = data.Addresses;
guidata(handles.Address_Book,handles)
%显示第一个条目
set(handles.Contact_Name,'String',data.Addresses(1).Name)
set(handles.Contact_Phone,'String',data.Addresses(1).Phone)
%将索引指针设置为1并保存句柄
handles.Index = 1;
guidata(handles.Address_Book,handles)
else
errordlg('Not a valid Address Book','Address Book Error')
end
```

步骤三：控制菜单回调行为：

首先看Open菜单项回调函数Check_And_load，该函数将确认并装载新的地址簿。

Open菜单项的回调函数代码如下：

```
function varargout = Open_Callback(h, eventdata, handles, varargin)
[filename, pathname] = uigetfile({'*.mat', 'All MAT-Files (*.mat)'; ...
    '*.*','All Files (*.*)'}, ...
    'Select Address Book');
% 如果选择Cancel则返回
if isequal([filename,pathname],[0,0])
return
% 否则构造文件全路径并检查装载该文件
else
File = fullfile(pathname,filename);
% 如果MAT文件无效，不保存信息
if Check_And_Load(File,handles)
    handles.LastFIle = File;
    guidata(h,handles)
end
end
```

再来看Save和Save as回调函数，该回调函数使用菜单项的Tag属性来判断究竟是Save还是Save as为回调对象，也就是说哪一个对象的句柄作为回调函数的第一个参数。如果用户选择的是Save菜单项，则调用save命令来保存MAT文件的新名字和电话号码；如果选择的是Save as，则显示一个指定保存目标文件名的对话框，用户可以选择一个已有的文件，也可以输入新的文件名，对话框将返回文件名和路径，这一步包括：

- 使用 fullfile 来创建一个与平台无关的路径名；
- 调用 save 来保存 MAT 文件中的新数据；

- 刷新句柄结构体以包含新的MAT文件名；
- 调用guidata保存句柄结构体。

Save_Callback回调函数代码如下：

```
function varargout = Save_Callback(h, eventdata, handles, varargin)
        % 获取被选菜单的Tag属性
        Tag = get(h,'Tag');
        %获取addresses数组
        Addresses = handles.Addresses;
        %根据所选项执行相应的行为
        switch Tag
        case 'Save'
        % 保存到缺省的地址簿文件中去
        File = handles.LastFile;
        save(File,'Addresses')
        case 'Save_As'
        % 允许用户选择要保存的目标文件名
        [filename, pathname] = uiputfile({'*.mat';'*.*'},'Save as');
        %如果选择Cancel则返回
        if isequal([filename,pathname],[0,0])
            return
        else
            % 构造全路径名并保存
            File = fullfile(pathname,filename);
            save(File,'Addresses')
            handles.LastFile = File;
            guidata(h,handles)
        end
        end
```

第三步来看Create New菜单。Create New菜单的回调函数主要将编辑框的String属性设置为空。其代码如下：

```
function varargout = New_Callback(h, eventdata, handles, varargin)
set(handles.Contact_Name,'String','')
set(handles.Contact_Phone,'String','')
```

步骤四：控制组件回调行为：

首先看Contact Name编辑框回调函数。回调函数使用句柄结构体来访问地址簿的内容并获得索引指针，这样回调函数就能够判断用户修改之前编辑框中显示的是什么姓名。索引指针表明当前显示的姓名位置，在发布GUI时由Check_And_Load函数来添加地址簿和索引指针。如果用户添加了一个新的条目，回调函数将新的姓名添加到地址簿中并刷新索引指针来显示新的数值。刷新后的地址簿和索引指针再一次被保存在句柄结构体中。Contact Name编辑框的回调函数代码如下：

```
function varargout = Contact_Name_Callback(h, eventdata, handles, varargin)
        %在Contact Name和Phone编辑框中获取字符串
        Current_Name = get(handles.Contact_Name,'string');
        Current_Phone = get(handles.Contact_Phone,'string');
        %如果为空则返回
        if isempty(Current_Name)
        return
        end
        %从句柄结构体中获取当前地址列表
        Addresses = handles.Addresses;
        %搜索姓名列表，判断是否与一个已有姓名相同
        for i = 1:length(Addresses)
        if strcmp(Addresses(i).Name,Current_Name)
            set(handles.Contact_Name,'string',Addresses(i).Name)
            set(handles.Contact_Phone,'string',Addresses(i).Phone)
            handles.Index = i;
            guidata(h,handles)
            return
        end
        end
        %如果是个新的姓名，请求创建一个新的条目
        Answer=questdlg('Do you want to create a new entry?',
         'Create New Entry', 'Yes','Cancel','Yes');
        switch Answer
        case 'Yes'
        Addresses(end+1).Name = Current_Name; % Grow array by 1
        Addresses(end).Phone = Current_Phone;
        index = length(Addresses);
        handles.Addresses = Addresses;
        handles.Index = index;
        guidata(h,handles)
        return
        case 'Cancel'
        %回复为初始数值
        set(handles.Contact_Name,'string',Addresses(handles.Index).Name)
        ntact_Phone,'String',Addresses(handles.Index).Phone)
        return
        end
```

再来看Contact Phone #回调函数。和Contact Name编辑框类似，这个回调函数使用索引指针来刷新地址簿中的新数值或恢复为上一次的显示内容。所有当前地址簿和索引指针都保

存在句柄结构体中，因而其他回调函数可以获得这些数据。以下是Contact Phone#编辑框的回调函数代码：

```
function varargout = Contact_Phone_Callback(h, eventdata, handles, varargin)
        Current_Phone = get(handles.Contact_Phone,'string');
        % 如果有一个为空则返回
        if isempty(Current_Phone)
        return
        end
        % 在句柄结构体中获取当前地址列表
        Addresses = handles.Addresses;
        Answer=questdlg('Do you want to change the phone number?', ...
                'Change Phone Number', 'Yes','Cancel','Yes');
        switch Answer
        case 'Yes'
        % 如果没有相匹配的名字则创建一个新的条目
        Addresses(handles.Index).Phone = Current_Phone;
        handles.Addresses = Addresses;
        guidata(h,handles)
        return
        case 'Cancel'
        % 回复为初始值
        set(handles.Contact_Phone,'String',Addresses(handles.Index).Phone)
        return
        end
```

最后看Prev和Next按钮的回调函数。回调函数定义一个额外的参数str来表明被点击的按钮是Prev还是Next。Prev按钮的回调字符串以Prev为最后一个参数，而Next按钮则以Next为最后一个参数。在case语句中用str的数值来区分并实现每一个按钮的功能。Prev_Next_Callback回调函数将从句柄结构体中获取当前的索引指针和地址，并且根据被选择的按钮来增加或减小索引指针的数值并显示相应的名字和电话号码，最后将在句柄结构体中存储新的索引指针数值并使用guidata来保存刷新后的结构体。

Prev_Next_Callback回调函数代码如下：

```
function varargout = Prev_Next_Callback(h,eventdata,handles,str)
        % 获取索引指针和地址
        index = handles.Index;
        Addresses = handles.Addresses;
        % 根据被点击的按钮修改显示结果
        switch str
        case 'Prev'
        % 索引减1
        i = index - 1;
```

```
%如果索引小于1，设置其为Addresses数组最后一个元素的索引
if i < 1
    i = length(Addresses);
end
case 'Next'
% 索引增1
i = index + 1;
%如果索引超出范围，设置其为Addresses数组第一个元素的索引
if i > length(Addresses)
    i = 1;
end
end
% 为被选索引获取相应数据
Current_Name = Addresses(i).Name;
Current_Phone = Addresses(i).Phone;
set(handles.Contact_Name,'string',Current_Name)
set(handles.Contact_Phone,'string',Current_Phone)
%刷新索引指针以反映新的索引
handles.Index = i;
guidata(h,handles)
```

步骤五：自定义重画函数。重画函数将对以下几种可能发生的情况进行处理：

- 改变宽度：如果新的宽度大于固有宽度，图形窗口将被设置为新的宽度。Contact Name 编辑框的尺寸随着窗口宽度的变化而变化。这一功能由以下操作实现：修改编辑框的 Units 为 normalized；重新设置编辑框的宽度为窗口宽度的 78.9%；将 Units 重新设置为 characters。如果新的宽度小于固有宽度，窗口被设置为固有宽度；
- 改变高度：如果用户试图修改窗口高度，高度始终被设置为固有高度。但是由于用户释放鼠标时会调用重画函数，所以重画函数不能总是判断 GUI 在屏幕中的故有位置。因此，重画函数给窗口垂直分量（Position 向量的第二个元素）如下的补偿：

鼠标释放时的垂直位置+鼠标释放时的高度-原始高度

当窗口从底部被重画时，窗口将保持原位；从顶部重画时，窗口将移动到鼠标释放处的位置。

重画函数调用movegui函数来确保无论用户在何处释放鼠标，重画后的窗口始终保持在屏幕中。第一次发布GUI时，GUI窗口的大小和位置由Position属性决定，用户可以使用属性检查器来设置这个属性。

重画函数代码如下：

```
function varargout = ResizeFcn(h, eventdata, handles, varargin)
    % 获取窗口尺寸和位置
    Figure_Size = get(h,'Position');
    % 设置窗口固有尺寸
    Original_Size = [ 0 0 94 19.230769230769234];
```

```
% 如果重画窗口小于固有窗口尺寸，实行补偿
if (Figure_Size(3)<Original_Size(3)) | (Figure_Size(4) ~= Original_Size(4))
if Figure_Size(3) < Original_Size(3)
    % 如果宽度过小则设置为固有宽度
    set(h,'Position',...
    [Figure_Size(1) Figure_Size(2) Original_Size(3) Original_Size(4)])
    Figure_Size = get(h,'Position');
end
if Figure_Size(4) ~= Original_Size(4)
    % 不允许修改高度
    set(h,'Position', [Figure_Size(1),
            Figure_Size(2)+Figure_Size(4)-Original_Size(4),...
            Figure_Size(3), Original_Size(4)])
end
end
% 设置Contact Name编辑框Units属性为Normalized'
set(handles.Contact_Name,'units','normalized')
% 获取位置
C_N_pos = get(handles.Contact_Name,'Position');
%重新设置宽度使之与窗口相匹配
set(handles.Contact_Name,'Position',...
    [C_N_pos(1) C_N_pos(2) 0.789 C_N_pos(4)])
% 将Units重新设置为Characters'
set(handles.Contact_Name,'units','characters')
%在屏幕中重新放置GUI
movegui(h,'onscreen')
```

9.5 小　　结

本章介绍了如何进行 GUI 回调函数编程从而控制 GUI 的交互行为。通过本章的学习，读者将会对回调函数的编程方法有一个较为深入的理解，掌握各种组件行为的控制方法。通过以后的 GUI 编程实践，相信读者能够完全自如地进行 GUI 编程。

第 10 章　GUI 应用实例

本章将通过向读者介绍几个实例来说明 GUI 的实现技术。读者将在本章中掌握创建一个 GUI 的完整过程并复习在以上章中学到的 GUI 创建技术。在本章的每一个实例中都介绍了相应的 GUIDE 和 M 文件的使用方法。

10.1　实例一：关闭询问对话框

本例将在用户的图形窗口中显示一个对话框进行询问与确认操作。本例中的对话框是一个模态对话框，只有用户响应该对话框后，MATLAB 才会继续运行并将询问结果返回给调用该对话框的 GUI。

当 GUI 用户选择关闭应用程序窗口时，用户希望能够通过显示一个确认对话框来防止用户无意中进行的 GUI 关闭操作。要求该对话框能够实现以下几个功能：

- 该对话框能够被应用程序的 Close 按钮激活；
- 询问用户是否确认关闭操作（yes 或 no）；
- 在用户未响应前阻止 MATLAB 的继续执行；
- 由于存在未决操作，故使该对话框模态化，维持输入焦点；
- 当用户对询问对话框不做响应而使用窗口管理器关闭对话框时，关闭操作应不予执行。

该 GUI 的外观如图 10-1 所示，可以在 MATLAB 环境下通过运行该 GUI 的 FIG 文件来打开这个 GUI 界面。下面讨论这个对话框的实现步骤。

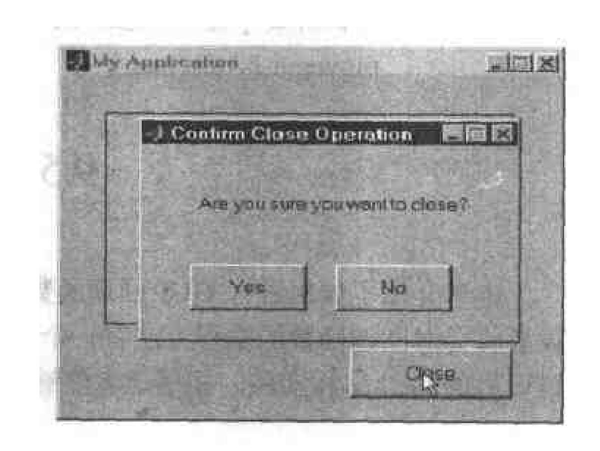


图 10-1　包含关闭询问对话框的 GUI 外观

10.1.1　GUI 组态

对调用关闭询问对话框的 GUI 来说，需要一个 Close 按钮，当用户点击该按钮准备退出 GUI 程序时，Close 按钮的回调函数首先需要发布一个用来确认即将执行关闭操作的对话框，然后这个回调函数必须等待该对话框返回一个数值。为了达到这个目的，该对话框的应用程序 M 文件将指定一个输出参数，Close 按钮的回调函数等待该参数返回。这也就是说，调用关闭询问对话框的 GUI 本身的界面设计工作仅仅包括布置一个 Close 按钮并进行属性设置，无需进行 GUI 组态。打开界面设计编辑器窗口，添加一个按钮，然后右击鼠标来显示文本菜单，选择 Properties Inspector 选项启

动属性检查器，在属性检查器中修改按钮的 Tag 属性和 Callback 属性。

对于关闭询问对话框来说，为了使对话框能够等待用户输入，在进行 GUI 组态时，要对该对话框进行组态。选择 GUIDE 应用程序选项对话框的 Function does not return until application window dismissed 选项。这个选项将给对话框的应用程序 M 文件添加一个 uiwait 调用命令。另外，为了使对话框模态化，使用属性检查器来设置窗口的 WindowStyle 属性值为 modal（参见图 10-2 所示）。

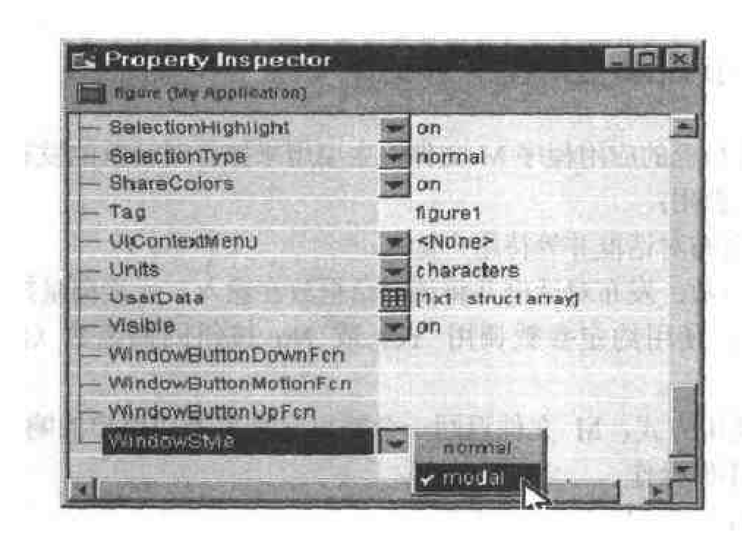


图 10-2 关闭询问对话框模态化设置

10.1.2 Close 按钮回调函数

当用户在 GUI 应用程序窗口中按下 Close 按钮后，以下行为将顺序地发生：

- 用户点击 Close 按钮：回调函数调用 M 文件来发布确认对话框并等待返回值；
- 执行确认对话框的 M 文件并等待用户执行三种可能的操作之一：选择 Yes 按钮、选择 No 按钮或关闭该对话框（使用系统 X 按钮）。其他所有交互行为均无效；
- 确认对话框通过将输出数据返回到 Close 按钮回调函数来恢复 M 文件的执行，Close 按钮的回调函数恢复执行并根据用户对确认对话框的操作方式来执行相应的操作。

于是 Close 按钮的回调函数必须按照以下步骤执行 GUI 的运行：

- 根据当前 GUI 的位置和大小确定确认对话框的位置；
- 通过使用一个用来指定对话框位置的输入参数以及一个导致回调函数等待对话框返回输出参数来调用确认对话框 M 文件；
- 根据确认对话框返回的答案执行相应的行为。

Close 回调函数代码如下：

```
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
        pos_size = get(handles.figure1,'Position');
        dlg_pos = [pos_size(1)+pos_size(3)/5 pos_size(2)+pos_size(4)/5];
        user_response = modaldlg(dlg_pos);
        switch user_response
        case {'no','cancel'}
```

```
return
case 'yes'
%准备关闭GUI应用程序窗口
    ...
delete(handles.figure1)
end
```

10.1.3 关闭询问对话框应用程序 M 文件

确认对话框有自己的应用程序 M 文件，主要用来执行对话框的发布。这个 M 文件可以使用以下三种方式调用：

- 无参数：发布对话框并等待用户输入；
- 一个数值参数：发布对话框并将该对话框放在输入二元素向量指定的位置处；
- 四个参数：使用约定参数调用 Yes 或 No 按钮回调函数（h, eventdata, handles, varargin）。

对于每一种调用方式，M 文件返回一个输出字符串表明用户的响应。下面介绍应用程序 M 文件执行的不同操作。

1. *发布对话框*

如果输入参数的个数是 0 或 1，则以下的应用程序 M 文件代码将执行对话框的发布工作，包括以下几个方面：

- 检查输入参数个数是否正确（回调函数有四个参数）；
- 使用 openfig 来装载 FIG 文件；
- 设置窗口颜色为主机标准 GUI 颜色；
- 创建句柄结构体。

注意函数将返回一个输出参数 answer 并将其传递给 Close 按钮的回调函数。

```
function answer = modaldlg(varargin)
    % 函数接受0，1，4各参数
    error(nargchk(0,4,nargin))
    if nargin == 0 | isnumeric(varargin{1})
    fig = openfig(mfilename,'reuse');
    set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
    handles = guihandles(fig);
    guidata(fig, handles);
```

2. *指定对话框的位置*

对话框应用程序 M 文件接受一个指定在何处显示对话框的输入参数，这就使得对话框应用程序 M 文件能够根据应用程序的窗口来放置对话框。这个输入参数是一个两元素的向量，包含对话框左边和底边相对于屏幕右下角的偏移量（以像素为单位）。这个向量由 Close 按钮回调函数给出。

某些情况下重新定位对话框的代码段在调用 set 命令之前将导致窗口的闪烁。为了防止

这种影响，在保存对话框时将对话框窗口的 Visible 属性值定义为 off，重新定义位置参数后再将该属性定义为 on。注意用户必须在设置 Visible 属性之前指定对话框的 Position 属性。

```
        if nargin == 1
    pos_size = get(fig,'Position');
    pos = varargin{1};
    if length(pos) ~= 2
        error('Input argument must be a 2-element vector')
    end
    new_pos = [pos(1) pos(2) pos_size(3) pos_size(4)];
    set(fig,'Position',new_pos,'Visible','on')
        end
```

3. 等待用户响应

uiwait 命令将导致模态对话框在返回 Close 按钮回调函数之前等待。在这期间对话框回调函数能够按照用户的选择执行操作。uiwait 函数将保持等待直到对话框窗口被删除或执行一个 uiresume 命令，也就是说以下情况将中止等待 uiwait 函数的执行：

- 用户点击系统边框的关闭按钮：在这种情况下 uiwait 函数将返回。由于此时存储在变量 fig 中的句柄不再与图形窗口一致，所以模态对话框使用 ishandle 函数来测试句柄以向 Close 按钮回调函数返回“cancel”；
- Yes 按钮回调函数在设置 handles.answer 为 yes 后执行 uiresume 命令；
- No 按钮回调函数在设置 andles.answer 为 no 后执行 uiresume 命令。

```
uiwait(fig);
if ~ishandle(fig)
    answer = 'cancel';
else
    handles = guidata(fig);
    answer = handles.answer;
delete(fig);
end
```

4. 执行回调函数

以下是使模态对话框能够执行回调子函数的调度函数代码。这些代码是建立在模态对话框被调用来执行一个回调函数的基础上的，第一个参数表示需要调用的回调函数名称的字符串：

```
elseif ischar(varargin{1}) %调用指定的子函数或回调函数
try
    [varargout{1:nargout}] = feval(varargin{:});
catch
disp(lasterr);
end
end
```

5. 定义 Yes 和 No 按钮的回调函数

Yes 和 No 按钮的回调函数同样地执行以下基本步骤：

- 在句柄结构体的 answer 域中指定用户的响应类型；
- 使用 guidata 来保存模态化的句柄结构体供以后主程序的查询；
- 使用 uiresume 来继续执行主程序被终止的代码。

每个按钮用户控件的 Tag 属性在保存应用程序 M 文件之前都会作相应修改，这就使得回调函数名更具描述性。以下代码说明了回调函数的执行过程：

```
function varargout = NoButton_Callback(h, eventdata, handles, varargin)
    handles.answer = 'No';
    guidata(h, handles);
    uiresume(handles.figure1);
function varargout = YesButton_Callback(h, eventdata, handles, varargin)
    handles.answer = 'Yes';
    guidata(h,handles);
    uiresume(handles.figure1);
```

10.1.4　使用关闭询问函数保护 GUI

无论用户何时关闭窗口，MATLAB 都会先执行由窗口 CloseRequestFcn 属性定义的关闭询问函数。缺省的关闭询问窗口将简单地删除窗口，然而，GUI 希望当用户还没有响应关闭询问对话框但无意中点击窗口边框的关闭按钮时能够保护 GUI 窗口不被删除。为此，用户可以通过重新设置 CloseRequestFcn 属性值来改变缺省的关闭询问函数。在介绍 Close 按钮的回调函数时已经说明了 GUI 的 Close 按钮回调函数同时能够作为一个关闭函数。为了在用户的应用程序 M 文件中添加新的关闭函数，选择界面设计编辑器中的窗口并右击，然后从文本菜单中选择 Edit CloseRequestFcn 选项（参见图 10-3）。GUIDE 将一个新的子函数自动添加到 GUI 的应用程序 M 文件中，同时修改窗口的 CloseRequestFcn 属性将被添加的函数作为关闭请求函数来执行。

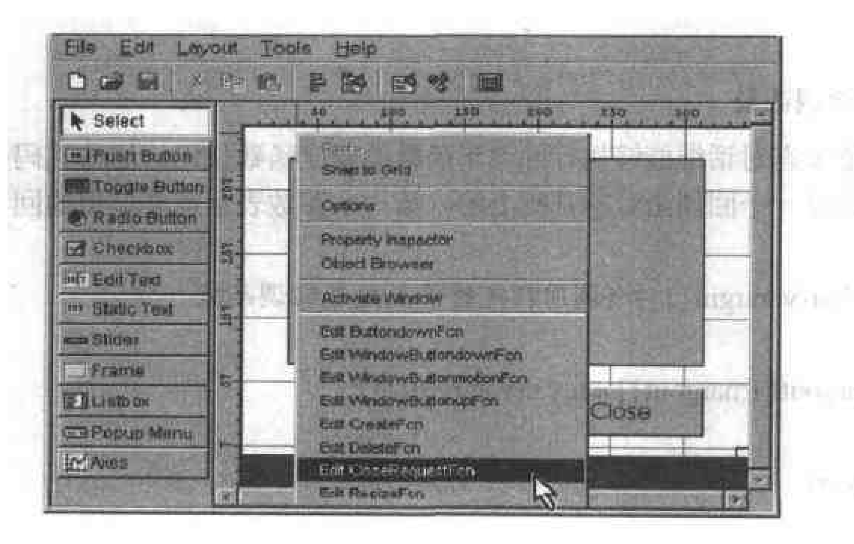


图 10-3　设置新的窗口关闭函数

GUI 关闭请求函数简单地调用 Close 按钮回调函数：

```
function varargout = figure1_CloseRequestFcn(h,eventdata,handles,varargin)
    pushbutton1_Callback(h,eventdata,handles)
```

10.1.5 M 文件代码

模态对话框 M 文件的全部代码如下：

```
function answer = modaldlg(varargin)
    error(nargchk(0,4,nargin))
    if nargin == 0 | isnumeric(varargin{1})
        fig = openfig(mfilename,'reuse');
        set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargin == 1
            pos_size = get(fig,'Position');
            pos = varargin{1};
            if length(pos) ~= 2
                error('Input argument must be a 2-element vector')
            end
            new_pos = [pos(1) pos(2) pos_size(3) pos_size(4)];
            set(fig,'Position',new_pos,'Visible','on')
            figure(fig)
            end
        uiwait(fig);
        if ~ishandle(fig)
            answer = 'cancel';
            else
                handles = guidata(fig);
                answer = handles.answer;
            delete(fig);
            end
        elseif ischar(varargin{1})
            try
                [varargout{1:nargout}] = feval(varargin{:});
            catch
                    disp(lasterr);
            end
    end
function varargout = NoButton_Callback(h, eventdata, handles, varargin)
```

```
    handles.answer = 'No';
    guidata(h, handles);
    uiresume(handles.figure1);
function varargout = YesButton_Callback(h, eventdata, handles, varargin)
    handles.answer = 'Yes';
    guidata(h,handles);
    uiresume(handles.figure1);
```

GUI 应用程序 M 文件代码如下：

```
function varargout = example1_close(varargin)
    if nargin == 0
            fig = openfig(mfilename,'reuse');
        set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
            varargout{1} = fig;
            end
        elseif ischar(varargin{1})
        try
            [varargout{1:nargout}] = feval(varargin{:});
            catch
            disp(lasterr);
            end
    end
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
    pos_size = get(handles.figure1,'Position');
    user_response = modaldlg(
    [pos_size(1)+pos_size(3)/5 pos_size(2)+pos_size(4)/5]);
    switch user_response
    case {'No','cancel'}
    case 'Yes'
        delete(handles.figure1)
    end
function varargout = figure1_CloseRequestFcn(h, eventdata, handles, varargin)
    pushbutton1_Callback(h,eventdata,handles)
```

10.2　实例二：路径列表框阅读器

本例将显示一个文件路径列表框，该列表框提供显示路径内容、浏览其他目录和定义用户双击操作命令的功能。当用户双击一个列表项时，如果该列表项是个文件，那么 GUI 将按照文件类型打开该文件；如果列表项是一个子目录，GUI 将该子路径下的文件读入列表框中；如果列表项是一个单点号(.)，GUI 将重新显示当前目录；如果列表项是一个双点号（..），GUI 将显示为上一级目录中的所有文件。

图 10-4 说明了该 GUI 的外观。

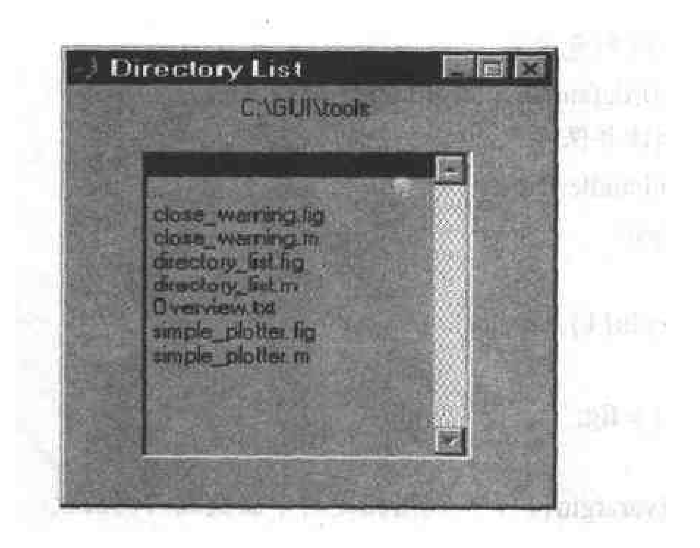


图 10-4　路径列表框阅读器外观

以下将介绍 GUI 的具体实现方法。

10.2.1　指定列表框目录

这部分内容主要说明在发布 GUI 时怎样将一个路径作为输入参数进行传递。

用户可以通过将一个完整的路径名作为字符串输入参数来指定 GUI 第一次显示时的路径。如果不指定路径（即用户调用应用程序 M 文件时不使用输入参数），GUI 使用 MATLAB 的当前路径。通常，如果调用应用程序 M 文件时没有输入参数，则应用程序 M 文件将发布 GUI，如果有一个字符串参数则调用字符串指定的子函数。本例对这些行为进行了修改使用户可以使用以下方式调用 M 文件：

- 无输入参数：使用 MATLAB 当前路径发布 GUI；
- 一个有效路径作为输入参数：发布 GUI 并显示指定路径；
- 参数个数多于一个并且第一个参数不是路径而是字符串：执行由参数指定的子函数（执行回调函数）。

以下代码列表是应用程序 M 文件的整个初始化部分的代码：

```
function varargout = lbox2(varargin)
```

```
if nargin <= 1 %发布GUI
if nargin == 0
      initial_dir = pwd;
elseif nargin == 1 & exist(varargin{1},'dir')
      initial_dir = varargin{1};
else
      errordlg('Input argument must be a valid directory',...
                  'Input Argument Error!')
      return
      end
fig = openfig(mfilename,'reuse');
% 为窗口使用系统颜色设置
set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
% 生成句柄结构体并保存
      handles = guihandles(fig);
guidata(fig, handles);
% 填充列表框
load_listbox(varargin{1},handles)
if nargout > 0
      varargout{1} = fig;
end
      elseif ischar(varargin{1}) % 调用指定的子函数或回调函数
      try
            [varargout{1:nargout}] = feval(varargin{:});
      catch
      disp(lasterr);
end
end
```

10.2.2 装载列表框

这部分内容描述了在列表框中装载目录内容的回调函数，这个函数还能够在句柄结构体中保存目录内容的相关信息。本例使用一个单独的子函数将列表项装载到列表框中，这个子函数以路径和句柄结构体为输入参数，可执行以下操作：

- 修改指定路径使 GUI 可以按照需要在路径树中漫游；
- 使用 dir 命令获取指定目录下的文件列表并判断一个名称代表的是子路径还是文件。dir 返回一个包含这些信息的结构体（dir_struct），该结构体包括 name 和 isdir 两个域；
- 将文件和路径名分类并在句柄结构体中保存分好类的名称和其他信息以便这些信息可以传递给其他函数。dir_struct 结构体的 name 域将作为一个单元数组传递给

sortrows，这个单元数组每一行将被转换为一个文件名，而 isfir 域及其索引将作为向量保存在句柄结构体中；

- 调用 guidata 来保存句柄结构体；
- 设置列表框的 String 属性以显示文件和路径名，同时将 Value 属性值设为 1。由于 MATLAB 在选择列表项时，而不是在 String 属性发生变化时改变 Value 属性，所以有必要保证 Value 属性不会超过 String 中列表项的范围；
- 通过设置 String 属性值为 pwd 命令的输出结果使当前路径显示在一个编辑框中。

load_listbox 函数将在应用程序 M 文件初始化阶段和列表框回调函数中被调用：

```
function load_listbox(dir_path,handles)
    cd (dir_path)
    dir_struct = dir(dir_path);
    [sorted_names,sorted_index] = sortrows({dir_struct.name}');
    handles.file_names = sorted_names;
    handles.is_dir = [dir_struct.isdir];
    handles.sorted_index = [sorted_index];
    guidata(handles.figure1,handles)
    set(handles.listbox1,'String',handles.file_names,'Value',1)
    set(handles.text1,'String',pwd)
```

10.2.3　列表框回调函数

这部分内容说明列表框是怎样编程实现用户在列表框中双击列表项时所响应的操作的。

列表框回调函数仅仅控制一种事件：列表项被双击。在列表框中双击一个列表项是打开文件的一种标准方式。如果被选择的列表项是个文件，那么该文件名将被传递给 open 函数；如果是一个路径，那么 GUI 将修改路径并显示该路径的内容。回调函数利用 open 命令能够控制多种类型的文件来工作，但对 FIG 文件则另当别论。对于 FIG 文件，回调函数并不打开该文件而是将它传递给 guide 命令进行编辑。若在试图打开一个文件的过程中发生了错误，那么该错误将会被一个错误对话框捕获，同时选中的文件也不会被打开。

由于单击列表项也会调用列表框回调函数，所以有必要询问窗口的 Selection Type 属性来判断用户是否执行了双击操作。列表项的双击操作将会把 Selection Type 属性设置为 open。列表框中的所有列表项用从 1 到 n 的下标来引用，其中 1 表示第一项，n 表示第 n 项。MATLAB 将这个索引保存在列表框的 Value 属性中。回调函数使用这个索引从 String 属性中包含的列表项中获取被选列表项的名称。

load_listbox 函数使用 dir 命令来获得用来说明列表项是文件还是子路径的数值列表。这些数值（1 代表是子路径，0 代表文件）被保存在句柄结构体中。列表框回调函数通过查询这些数值来判断当前选中的列表项是文件还是子路径。如果被选项是子路径，那么将切换到所选路径下并再次调用 load_listbox 函数将列表框的内容换成新路径的内容；如果被选项是文件，那么就通过获取文件的扩展名（filepart）来判断该文件是否为 FIG 文件，如果为 fig，则使用 guide 命令来打开它，其他类型的文件则传递给 open 命令。open 语句在一个 try/catch 块中调用，这样就使得打开文件时发生的错误能够传递给错误对话框而不是命令行。

列表框回调函数代码如下：

```
function varargout = listbox1_Callback(h,eventdata,handles,varargin)
    if strcmp(get(handles.figure1,'SelectionType'),'open')
    index_selected = get(handles.listbox1,'Value');
    file_list = get(handles.listbox1,'String');
    filename = file_list{index_selected};
    if handles.is_dir(handles.sorted_index(index_selected))
        cd (filename)
        load_listbox(pwd,handles)
    else
        [path,name,ext,ver] = fileparts(filename);
        switch ext
        case '.fig'
            guide (filename)
            otherwise
            try
                open(filename)
            catch
            errordlg(lasterr,'File Type Error','modal')
            end
        end
    end
    end
```

通过创建一个名为 namexyz 的 M 文件（xyz 为扩展名），用户可以打开所有 open 命令能够识别的、扩展名为三个字符的文件类型。注意列表框回调函数并不使用这个方法来打开 FIG 文件（即后缀为 fig 的文件）的，此时应用程序 M 文件需要 openfig.m 这个 M 文件。

10.2.4 应用程序 M 文件全部代码

```
function varargout = lbox2(varargin)
    if nargin <= 1
        if nargin == 0
            initial_dir = pwd;
        elseif nargin == 1 & exist(varargin{1},'dir')
            initial_dir = varargin{1};
        else
            errordlg('Input argument must be a valid directory',...
                    'Input Argument Error!')
            return
        end
```

```
        fig = openfig(mfilename,'reuse');
    handles = guihandles(fig);
        guidata(fig, handles);
        load_listbox(initial_dir,handles)
        if nargout > 0
            varargout{1} = fig;
        end
    elseif ischar(varargin{1})
        try
            [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
            catch
            disp(lasterr);
            end
    end
function varargout = listbox1_Callback(h, eventdata, handles, varargin)
    if strcmp(get(handles.figure1,'SelectionType'),'open')
        index_selected = get(handles.listbox1,'Value');
        file_list = get(handles.listbox1,'String');
        filename = file_list{index_selected};
        if  handles.is_dir(handles.sorted_index(index_selected))
            cd (filename)
            load_listbox(pwd,handles)
        else
            [path,name,ext,ver] = fileparts(filename);
            switch ext
            case '.fig'
                guide (filename)
            otherwise
                try
                open(filename)
                catch
                    errordlg(lasterr,'File Type Error','modal')
                end
            end
            end
function load_listbox(dir_path,handles)
    cd (dir_path)
    dir_struct = dir(dir_path);
    [sorted_names,sorted_index] = sortrows({dir_struct.name}');
    handles.file_names = sorted_names;
```

```
handles.is_dir = [dir_struct.isdir];
handles.sorted_index = [sorted_index];
guidata(handles.figure1,handles)
set(handles.listbox1,'String',handles.file_names,      'Value',1)
set(handles.text1,'String',pwd)
```

10.3　实例三：设置 Simulink 模型参数

simulink 是 MATLAB 的最为有效的仿真工具之一，本例说明了如何创建能够设置 simulink 模型参数的 GUI。另外，这个 GUI 可以运行仿真程序并绘制仿真结果。图 10-5 说明了 GUI 使用不同参数的增益控制器运行的三个仿真结果。

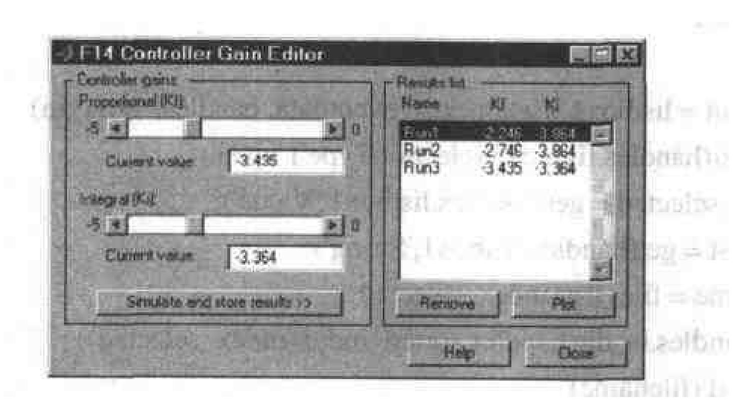


图 10-5　设置 Simulink 模型参数 GUI 的外观

本例中使用了以下几种 GUI 创建技术：

- 在 GUI 中打开并设置 simulink 模型的参数；
- 实现与编辑框联合操作的、既显示当前数值又接受用户输入的滚动条；
- 使用句柄结构体管理全局数据；
- 将图形输出导入句柄隐藏的图形窗口中；
- 添加一个在 MATLAB 帮助浏览器中显示 Html 文件的帮助按钮。

下面介绍该 GUI 的实现方法。

10.3.1　GUI 说明

MATLAB 提供一个 Simulink 例程：F14 飞机控制器增益设计器 simulink 程序 f14.mdl 来分析改变比例积分控制器采用的增益对飞机的攻击角度和飞行员能够感受到的重力产生的影响，本例将使用这个程序进行参数设置。注意在运行 GUI 时必须打开 f14.mdl，如果该程序被关闭，GUI 在需要执行该模型时将会重新打开它。

用户可以修改两个模块的增益大小：比例环节模块的增益 Kf 和积分环节传递函数模块的增益 Ki。可以通过以下两种方式之一修改增益：移动与增益相关的滚动条；在与增益相关联的 Current value 编辑框中键入新的增益值。模型中相应的数值将被重新显示为 GUI 中

输入的新值。

用户设置完增益值后就可以通过使用 Simulate and store results 按钮来运行仿真程序。仿真时间和输出变量将存储在 Results list 列表中。通过选择 Result lists 列表的某行并点击 plot 按钮来生成一个或多个仿真结果的图形。如果用户选择了多行，绘图结果将包含每一个仿真结果的图形。绘图结果显示在一个图形窗口中，用户每点击一次 plot 按钮将对该窗口进行一次清除。窗口的句柄被隐藏，从而保证只有该 GUI 才能在此窗口中绘制图形。

如果要在 Result lists 中删除一个仿真结果，选择该行（或多行）并按下 Remove 按钮。

10.3.2　发布 GUI

由于本例中的 GUI 是用来作为分析工具的，所以将该 GUI 设计为不中断、非模态的。GUI 使用以下应用程序选项：

- 重画行为：Non-resizable；
- 命令行可访问性：Off；
- 选择既生成 FIG 文件又生成 M 文件；
- 应用程序 M 文件选项选择：
  生成回调函数原型：Generate callback function prototypes；
  只允许一个实例运行：Application allows only one instance to run；
  使用系统背景颜色设置：Use system color scheme for background。

10.3.3　打开 simulink 模块流程

本例是设计用来操作 F14 仿真模型的。由于 GUI 将设置模型参数并运行仿真程序，所以显示 GUI 时要打开 F14 模型。当应用程序 M 文件发布 GUI 时将执行 model_open 函数，使用该函数的目的是：

- 判断模型是否打开，如果模型没有打开，则打开需要设置参数的模块流程和子系统；
- 修改控制器增益模块的大小以显示新设置的增益；
- 将 GUI 带到前台使之位于 simulink 流程图之上；
- 设置模块参数使之与 GUI 的当前设置相匹配。

以下是 model_open 子函数的代码：

```
function model_open(handles)
    if isempty(find_system('Name','f14')),
    open_system('f14'); open_system('f14/Controller')
    set_param('f14/Controller/Gain','Position',[275 14 340 56])
    figure(handles.F14ControllerEditor)
    set_param('f14/Controller Gain','Gain',...
    get(handles.KfCurrentValue,'String'))
    set_param('f14/Controller/Proportional plus integral compensator',...
            'Numerator',get(handles.KiCurrentValue,'String'))
    end
```

10.3.4 滚动条和编辑框编程

GUI 在设计时使用了一个有效的联合组件：每一个滚动条与一个编辑框相耦合使得编辑框能够显示滚动条的当前数值，用户可以在编辑框中键入数据，同时滚动条将按照新的数值自动重新定位。

GUI 使用两个滚动条指定模块的增益，这两个滚动条使得增益的数值能够在某个特定范围内连续变化。当用户改变滚动条的数值时，回调函数将执行以下操作：

- 调用 model_open 来确保 simulink 模型已打开从而使模型参数可以设置；
- 获取新的滚动条数值；
- 设置编辑框组件的当前取值与滚动条相匹配；
- 设置相应的模块参数为新的数值。

以下是 Proportional 滚动条（Kf 增益）回调函数的代码：

```
function varargout = KfValueSlider_Callback(h,eventdata,handles,varargin)
    % 确保模型已经打开
    model_open(handles)
    % 在滚动条中设置新的增益值
    NewVal = get(h,'Value');
    % 将Kf编辑框的当前取值设置为滚动条设置的新值
    set(handles.KfCurrentValue,'String',NewVal)
    % 设置Kf增益模块的增益参数为新的数值
    set_param('f14/Controller/Gain','Gain',num2str(NewVal))
```

KfCurrent value 编辑框使用户能够分别给每个参数设置数值，当用户在编辑框键入新值后点击其他组件时，编辑框回调函数将执行以下操作：

- 调用 model_open 来确保 simulink 模型已打开从而仿真参数可以设置；
- 将编辑框 String 属性返回的字符串转换为一个双精度数值（str2double）；
- 检查用户输入的参数是否在滚动条的范围内，如果数值超出范围，编辑框的 String 属性将设置为滚动条的数值，从而拒绝用户输入的数字；如果没有超出范围则滚动条的数值将刷新；
- 设置相应的模块参数为新的数值。

以下是 Kf Current value 编辑框回调函数的代码：

```
function varargout = KfCurrentValue_Callback(h,eventdata,handles,varargin)
    model_open(handles)
    % 获取Kf增益的新取值
    NewStrVal = get(h,'String');
    NewVal = str2double(NewStrVal);
    %检查输入数值是否在允许的范围之内
    if isempty(NewVal) | (NewVal< -5) | (NewVal>0),
    %恢复为滚动条表明的Kf上次取值
    OldVal = get(handles.KfValueSlider,'Value');
```

```
set(h,'String',OldVal)
else % 使用新的Kf数值
%将Kf滚动条的数值设置为新值
set(handles.KfValueSlider,'Value',NewVal)
%设置Kf增益模块的增益参数为新的取值
set_param('f14/Controller/Gain','Gain',NewStrVal)
end
```

10.3.5 在 GUI 中运行仿真程序

GUI的Simulink and store results按钮回调函数将对模型进行仿真并将仿真结果存储在句柄结构体中。由于句柄结构体是作为参数在函数间传递的，所以将数据存储在句柄结构体中将会简化向其他函数传递数据的过程。当用户点击Simulink and store results按钮时，其回调函数将执行以下操作：

- 调用 sim 函数来运行仿真并返回用来绘图的数据；
- 创建一个保存仿真结果的结构体；
- 在句柄结构体中保存仿真结果结构体；
- 刷新列表框的 String 来显示最新的运行。

以下是Simulink and store results按钮的回调函数代码：

```
function varargout = SimulateButton_Callback(h,eventdata,handles,varargin)
    [timeVector,stateVector,outputVector] = sim('f14');
    % 找回旧的结果数据结构体
    if isfield(handles,'ResultsData') & ~isempty(handles.ResultsData)
    ResultsData = handles.ResultsData;
    %判断当前使用的最大运行数目
    maxNum = ResultsData(length(ResultsData)).RunNumber;
    ResultNum = maxNum+1;
    else
    %建立结果数据结构体
    ResultsData = struct('RunName',[],'RunNumber',[],...
    'KiValue',[],'KfValue',[],'timeVector',[],'outputVector',[]);
    ResultNum = 1;
    end
    if isequal(ResultNum,1),
    % 使能plot和remove按钮
    set([handles.RemoveButton,handles.PlotButton],'Enable','on')
    end
    % 获取Ki和Kf数值，保存为数据并放入结果列表中。
    Ki = get(handles.KiValueSlider,'Value');
    Kf = get(handles.KfValueSlider,'Value');
```

```
ResultsData(ResultNum).RunName = ['Run',num2str(ResultNum)];
ResultsData(ResultNum).RunNumber = ResultNum;
ResultsData(ResultNum).KiValue = Ki;
ResultsData(ResultNum).KfValue = Kf;
ResultsData(ResultNum).timeVector = timeVector;
ResultsData(ResultNum).outputVector = outputVector;
%建立列表框的新结果列表字符串
ResultsStr = get(handles.ResultsList,'String');
if isequal(ResultNum,1)
ResultsStr = {['Run1 ',num2str(Kf),' ',num2str(Ki)]};
else
ResultsStr = [ResultsStr;...
        {['Run',num2str(ResultNum),' ',num2str(Kf),' ',num2str(Ki)]}];
end
set(handles.ResultsList,'String',ResultsStr);
%保存新的结果数据
handles.ResultsData = ResultsData;
guidata(h,handles)
```

10.3.6 在列表框中删除结果

GUI的Remove按钮回调函数将删除任何Result list列表框的被选中项。该函数还将删除句柄结构体中相应的运行数据。当用户点击Remove按钮时，回调函数执行以下操作：

- 判断用户点击 Remove 按钮时哪个列表项被选中，并将选中项从列表框的 String 属性中删除（将 String 中的对应项设置为空矩阵[]）；
- 删除句柄结构体中的相应数据；
- 如果所有项都被删除，显示字符串<empty>并使用 Enable 属性将 Remove 和 Plot 按钮设为无效；
- 使用 guidata 将修改结果保存在句柄结构体中。

以下是Remove按钮的回调函数：

```
function varargout = RemoveButton_Callback(h, eventdata, handles, varargin)
currentVal = get(handles.ResultsList,'Value');
resultsStr = get(handles.ResultsList,'String');
numResults = size(resultsStr,1);
%删除被选数值的数据和列表项
resultsStr(currentVal) =[];
handles.ResultsData(currentVal)=[];
%若无选项存在，将Remove和Plot按钮设为无效并修改列表字符串为empty
if isequal(numResults,length(currentVal)),
resultsStr = {'<empty>'};
```

```
currentVal = 1;
set([handles.RemoveButton,handles.PlotButton],'Enable','off')
end
% 确保列表框数据有效，然后重新设置Value和String
currentVal = min(currentVal,size(resultsStr,1));
set(handles.ResultsList,'Value',currentVal,'String',resultsStr)
% 保存新的结果数据
guidata(h,handles)
```

10.3.7 绘制结果数据

GUI的Plot按钮回调函数创建一个返回数据的图形并添加一个图例。绘图数据由句柄结构体传递给回调函数，句柄结构体中还包含仿真运行时的增益设置。当用户点击Plot按钮时，回调函数执行以下操作：

- 为 Result list 中的被选中运行的项收集数据，包括时间和输出两个变量以及每个待绘制的运行结果的颜色；
- 根据保存的数据生成图例字符串；
- 创建绘图窗口和坐标轴并保存句柄以供 Close 按钮回调函数使用；
- 绘制数据，添加图例并使图形窗口可见。

绘图窗口在创建时是不可见的，一旦图形和图例被创建，该窗口就被设置为可见。为了避免该窗口成为命令行或其他GUI的输出对象，将其HandleVisibility和IntegerHandle属性设置为off。然而这将意味着图形窗口对plot和legend命令而言也是隐藏的。使用以下步骤在一个隐藏窗口中绘图：

- 在创建窗口时保存其句柄；
- 创建坐标轴，将其父对象设置为图形窗口的句柄，同时保存坐标轴句柄；
- 创建图形（一个或多个线条对象），保存这些线条对象的句柄，设置其父对象为坐标轴的句柄；
- 使窗口可见。

以下是Plot按钮回调函数的代码：

```
function varargout = PlotButton_Callback(h, eventdata, handles, varargin)
    currentVal = get(handles.ResultsList,'Value');
    % 获取绘图数据并根据指定颜色生成命令字符串
    legendStr = cell(length(currentVal),1);
    plotColor = {'b','g','r','c','m','y','k'};
    for ctVal = 1:length(currentVal);
    PlotData{(ctVal*3)-2} =
        handles.ResultsData(currentVal(ctVal)).timeVector;
    PlotData{(ctVal*3)-1} =
        handles.ResultsData(currentVal(ctVal)).outputVector;
    numColor = ctVal - 7*( floor((ctVal-1)/7) );
```

```
        PlotData{ctVal*3} = plotColor{numColor};
        legendStr{ctVal} = andles.ResultsData(currentVal(ctVal)).RunName,...
            '; Kf=', ...
            num2str(handles.ResultsData(currentVal(ctVal)).KfValue),...
            '; Ki=',...
            num2str(handles.ResultsData(currentVal(ctVal)).KiValue)];
    end
        %如果有必要，创建绘图窗口并将其句柄保存在句柄结构体中
        if ~isfield(handles,'PlotFigure') | ~ishandle(handles.PlotFigure)
        handles.PlotFigure = figure('Name','F14 Simulation Output',...
            'Visible','off','NumberTitle','off',...
            'HandleVisibility','off','IntegerHandle','off');
        handles.PlotAxes = axes('Parent',handles.PlotFigure);
        guidata(h,handles)
        end
        % 绘制数据
        pHandles = plot(PlotData{:},'Parent',handles.PlotAxes);
        %添加图例并将窗口前移
        legend(pHandles(1:2:end),legendStr{:})
        % 使窗口可见并将其带到前台
        figure(handles.PlotFigure)
```

10.3.8　GUI 帮助按钮

GUI的Help按钮回调函数将在MATLAB帮助浏览器中显示一个HTML文件。该函数使用两个命令：当文件位于MATLAB路径中时返回文件全路径的which命令以及在帮助浏览器中显示文件的web命令。

以下代码是Help按钮的回调函数：

```
function varargout = HelpButton_Callback(h,eventdata,handles,varargin)
    HelpPath = which('f14ex_help.html');
    web(HelpPath);
```

用户还可以在网络浏览器中显示帮助文档或装载一个外部URL。

10.3.9　关闭 GUI

GUI的Close按钮回调函数将首先关闭绘图窗口（如果存在），然后关闭GUI。绘图窗口的句柄和GUI窗口都可以从句柄结构体中获得。回调函数执行以下操作步骤：

（1）检查句柄结构体中是否存在PlotFigure域，该域是否包含一个有效的窗口句柄（用户可以手动关闭图形窗口）；

（2）关闭GUI窗口。

以下是Close按钮回调函数的代码：

```
function varargout = CloseButton_Callback(h,eventdata,handles,varargin)
    % 关闭GUI以及任何打开的绘图窗口
    if isfield(handles,'PlotFigure') & ishandle(handles.PlotFigure),
        close(handles.PlotFigure);
    end
    close(handles.F14ControllerEditor);
```

10.3.10 列表框回调函数

由于列表项的操作行为都是由按钮来控制的，所以GUI不使用列表框回调函数。然而，当用户添加一个列表项时，GUIDE将自动插入一个回调函数原型并将设置callback属性使用户选择列表项时将调用自动生成的回调函数。本例无需执行列表框的回调函数，因而将其在应用程序M文件中删除。注意要同时删除Callback属性使得MATLAB不再试图执行回调函数。用户可以使用属性检查器来完成这项工作（参见图10-6）。

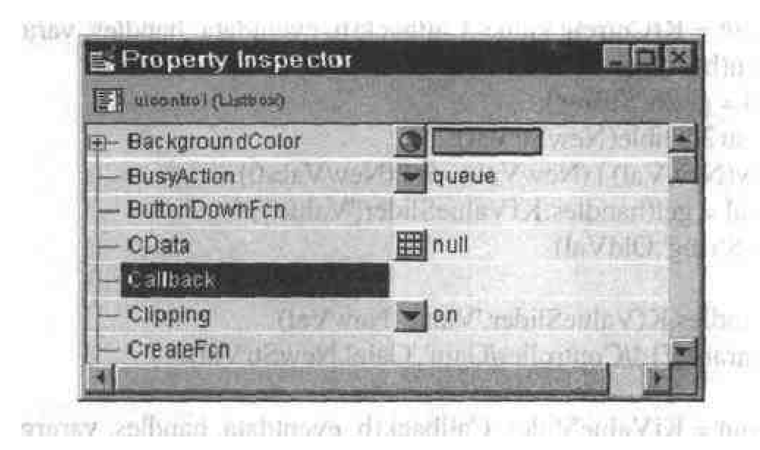


图 10-6 删除列表框回调函数的设置

10.3.11 应用程序 M 文件全部代码

```
function varargout = f14ex(varargin)
    if nargin == 0
        fig = openfig(mfilename,'reuse');
        set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
        handles = guihandles(fig);
        guidata(fig, handles);
        model_open(handles)
        if nargout > 0
            varargout{1} = fig;
            end
    elseif ischar(varargin{1})
        try
            [varargout{1:nargout}] = feval(varargin{:});
        catch
```

```
                    disp(lasterr);
                    end
        end
function model_open(handles)
        if   isempty(find_system('Name','f14')),
             open_system('f14');
             open_system('f14/Controller')
             set_param('f14/Controller/Gain','Position',[275 14 340 56])
             figure(handles.F14ControllerEditor)
             set_param('f14/Controller/Gain','Gain',...
             get(handles.KfCurrentValue,'String'))
             set_param('f14/Controller/Proportional plus integral compensator',...
                        'Numerator',get(handles.KiCurrentValue,'String'))
        end
function varargout = KfValueSlider_Callback(h, eventdata, handles, varargin)
        model_open(handles)
        NewVal = get(h,'Value');
        set(handles.KfCurrentValue,'String',NewVal)
        set_param('f14/Controller/Gain','Gain',num2str(NewVal))
function varargout = KfCurrentValue_Callback(h, eventdata, handles, varargin)
        model_open(handles)
        NewStrVal = get(h,'String');
        NewVal = str2double(NewStrVal);
        if   isempty(NewVal) | (NewVal< -5) | (NewVal>0),
             OldVal = get(handles.KfValueSlider,'Value');
             set(h,'String',OldVal)
        else
             set(handles.KfValueSlider,'Value',NewVal)
             set_param('f14/Controller/Gain','Gain',NewStrVal)
        end
function varargout = KiValueSlider_Callback(h, eventdata, handles, varargin)
        model_open(handles)
        NewVal = get(h,'Value');
        set(handles.KiCurrentValue,'String',NewVal)
        set_param('f14/Controller/Proportional plus integral compensator',...
                  'Numerator',num2str(NewVal))
function varargout = KiCurrentValue_Callback(h, eventdata, handles, varargin)
        model_open(handles)
            NewStrVal = get(h,'String');
            NewVal = str2double(NewStrVal);
        if   isempty(NewVal) | (NewVal< -5) | (NewVal>0),
             OldVal = get(handles.KiValueSlider,'Value');
             set(h,'String',OldVal)
        else
             set(handles.KiValueSlider,'Value',NewVal)
             set_param('f14/Controller/Proportional plus integral compensator',...
                        'Numerator',NewStrVal)
        end
function varargout = SimulateButton_Callback(h, eventdata, handles, varargin)
        [timeVector,stateVector,outputVector] = sim('f14');
```

```
        if isfield(handles,'ResultsData') & ~isempty(handles.ResultsData)
            ResultsData = handles.ResultsData;
            maxNum = ResultsData(length(ResultsData)).RunNumber;
            ResultNum = maxNum+1;
        else
            ResultsData = struct('RunName',[],'RunNumber',[],...
                    'KiValue',[],'KfValue',[],'timeVector',[],'outputVector',[]);
            ResultNum = 1;
        end
        if isequal(ResultNum,1),
            set([handles.RemoveButton,handles.PlotButton],'Enable','on')
        end
        Ki = get(handles.KiValueSlider,'Value');
        Kf = get(handles.KfValueSlider,'Value');
            ResultsData(ResultNum).RunName = ['Run',num2str(ResultNum)];
            ResultsData(ResultNum).RunNumber = ResultNum;
            ResultsData(ResultNum).KiValue = Ki;
            ResultsData(ResultNum).KfValue = Kf;
            ResultsData(ResultNum).timeVector = timeVector;
            ResultsData(ResultNum).outputVector = outputVector;
            ResultsStr = get(handles.ResultsList,'String');
            if isequal(ResultNum,1)
            ResultsStr = {['Run1                ',num2str(Kf),'      ',num2str(Ki)]};
            else
            ResultsStr = [ResultsStr;{['Run',num2str(ResultNum),'                ',...
                num2str(Kf),'      ',num2str(Ki)]}];
        end
        set(handles.ResultsList,'String',ResultsStr);
            handles.ResultsData = ResultsData;
            guidata(h,handles)
    function varargout = RemoveButton_Callback(h, eventdata, handles, varargin)
        currentVal = get(handles.ResultsList,'Value');
            resultsStr = get(handles.ResultsList,'String');
            numResults = size(resultsStr,1);
        resultsStr(currentVal) =[];
        handles.ResultsData(currentVal)=[];
        if isequal(numResults,length(currentVal)),
            resultsStr = {'<empty>'};
            currentVal = 1;
            set([handles.RemoveButton,handles.PlotButton],'Enable','off')
        end
        currentVal = min(currentVal,size(resultsStr,1));
        set(handles.ResultsList,'Value',currentVal,'String',resultsStr)
        guidata(h,handles)
    function varargout = PlotButton_Callback(h, eventdata, handles, varargin)
        currentVal = get(handles.ResultsList,'Value');
        legendStr = cell(length(currentVal),1);
            plotColor = {'b','g','r','c','m','y','k'};
            for ctVal = 1:length(currentVal);
            PlotData{(ctVal*3)-2} = handles.ResultsData(currentVal(ctVal)).timeVector;
```

```
            PlotData{(ctVal*3)-1} = handles.ResultsData(currentVal(ctVal)).outputVector;

        numColor = ctVal - 7*( floor((ctVal-1)/7) );
        PlotData{ctVal*3} = plotColor{numColor};
        legendStr{ctVal} = ...
            [handles.ResultsData(currentVal(ctVal)).RunName,';   Kf=', ...
        num2str(handles.ResultsData(currentVal(ctVal)).KfValue),';   Ki=', ...
        num2str(handles.ResultsData(currentVal(ctVal)).KiValue)];
    end
    if ~isfield(handles,'PlotFigure') | ~ishandle(handles.PlotFigure),
            handles.PlotFigure = figure('Name','F14 Simulation Output','Visible','off',...
            'NumberTitle','off','HandleVisibility','off','IntegerHandle','off');
            handles.PlotAxes = axes('Parent',handles.PlotFigure);
            guidata(h,handles)
    end
    pHandles = plot(PlotData{:},'Parent',handles.PlotAxes);
    legend(pHandles(1:2:end),legendStr{:})
    figure(handles.PlotFigure)
function varargout = HelpButton_Callback(h, eventdata, handles, varargin)
    HelpPath = which('f14ex_help.html');
    web(HelpPath);
function varargout = CloseButton_Callback(h, eventdata, handles, varargin)
    if isfield(handles,'PlotFigure') & ishandle(handles.PlotFigure),
        close(handles.PlotFigure);
    end
    close(handles.F14ControllerEditor);
```

10.4 实例四：从列表框访问工作平台变量

本例将在一个 GUI 中列举 MATLAB 基本工作平台中存在的变量并绘制这些变量。本例说明了如何选择多个列表项以及如何在不同的工作平台中执行绘图命令。本例中的 GUI 使用一个列表框来显示工作平台变量，用户可以使用这些变量来绘图。本例中将说明如何完成以下工作：

- 将列表框的内容显示为基本工作平台中存在的变量名；
- 初始化时不显示任何列表项；
- 允许在列表框中选择多个列表项；
- 当用户按下按钮时重新显示列表项；
- 在基本工作平台中执行绘图命令。

图 10-7 说明了这个 GUI 的外观。

注意本例中没有使用列表框的回调函数，这是由于绘图行为将由按钮来初始化，无需再使用列表框的回调函数。在这种情况下用户必须将空列表框回调函数留在应用程序 M 文件中或删除由列表框 Callback 属性定义的回调字符串。

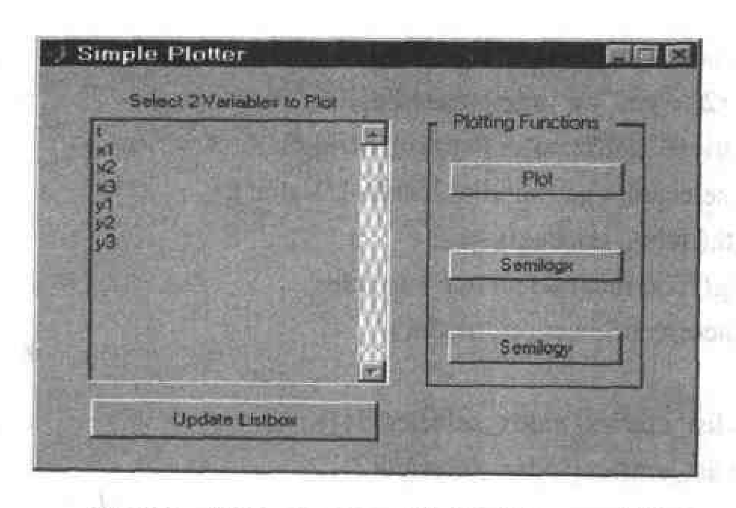


图 10-7 访问 MATLAB 工作平台变量 GUI 的外观

下面介绍该 GUI 的实现方法。

10.4.1 读取工作平台变量

在 GUI 进行初始化时需要查询工作平台变量并设置列表框的 String 属性，使列表框能够显示这些变量的名称。以下是添加到应用程序 M 文件中的子函数的代码，该子函数通过使用 evalin 函数在基本工作平台中执行 who 命令，从而实现上述功能。who 命令将返回一个用来填充列表框的字符串单元数组：

```
function update_listbox(handles)
vars = evalin('base','who');
set(handles.listbox1,'String',vars)
```

这个子函数的输入参数是应用程序 M 文件生成的句柄结构体，该结构体包括列表框的句柄以及 GUI 中其他组件的句柄。Updata Listbox 按钮的回调函数同样也调用函数 updata_listbox。

10.4.2 读取列表框的被选项

本例中的 GUI 允许用户在工作平台中选择两个变量，然后选择三个绘图函数（plot、semilogx 和 semilogy）中的一个来创建一幅图形。

为了实现在列表框中选择多个选项，用户必须设置 Min 和 Max 属性使 Max 与 Min 的差值大于 1。列表框多个选项的选择方法遵循大多数系统的标准方法：Control+鼠标左键可以用来选择多个不连续的选项；Shift+鼠标左键则用来选择多个连续选项。

当用户点击三个绘图函数按钮之一时，get_var_names 子函数将返回被选的两个变量名。该子函数将实现以下功能：

- 从 String 属性中获得列表框中所有列表项的列表；
- 在 Value 属性中获得被选项的索引号；
- 如果有两个选项被选择就返回两个字符变量，否则 get_var_names 将显示一个错误对话框提示用户必须选择两个变量。

以下是 ger_var_names 函数的代码：

```
function [var1,var2] = get_var_names(handles)
        list_entries = get(handles.listbox1,'String');
        index_selected = get(handles.listbox1,'Value');
        if length(index_selected) ~= 2
        errordlg('You must select two variables',...
            'Incorrect Selection','modal')
        else
        var1 = list_entries{index_selected(1)};
        var2 = list_entries{index_selected(2)};
        end
```

10.4.3　绘图按钮的回调函数

绘图按钮的回调函数调用 get_var_names 函数来获得待绘制变量的名称，然后调用 evalin 函数在基本工作平台中执行绘图命令。例如，以下是 plot 按钮的回调函数：

```
function varargout = plot_button_Callback(h,eventdata,handles,varargin)
    [x,y] = get_var_names(handles);
    evalin('base',['plot(' x ',' y ')'])
```

evalin 函数的参数由字符串和变量联合构成。

10.4.4　应用程序 M 文件全部代码

```
function varargout = lb(varargin)
    if nargin == 0
        % 发布GUI
        fig = openfig(mfilename,'reuse');
        set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
        handles = guihandles(fig);
        guidata(fig, handles);
        update_listbox(handles)
        set(handles.listbox1,'Value',[])
        if nargout > 0
            varargout{1} = fig;
        end
    elseif ischar(varargin{1})
        try
                [varargout{1:nargout}] = feval(varargin{:});
            catch
            disp(lasterr);
            end
    end
function varargout = update_button_Callback(h, eventdata, handles, varargin)
    update_listbox(handles)
```

```
function update_listbox(handles)
    vars = evalin('base','who');
    set(handles.listbox1,'String',vars)
function [var1,var2] = get_var_names(handles)
    list_entries = get(handles.listbox1,'String');
    index_selected = get(handles.listbox1,'Value');
    if length(index_selected) ~= 2
    errordlg('You must select two variables','Incorrect Selection','modal')
    else
    var1 = list_entries{index_selected(1)};
    var2 = list_entries{index_selected(2)};
    end
function varargout = plot_button_Callback(h, eventdata, handles, varargin)
    [x,y] = get_var_names(handles);
    evalin('base',['plot(',x,',',y,')'])
function varargout = semilogx_button_Callback(h, eventdata, handles, varargin)
    [x,y] = get_var_names(handles);
    evalin('base',['semilogx(',x,',',y,')'])
function varargout = semilogy_button_Callback(h, eventdata, handles, varargin)
    [x,y] = get_var_names(handles);
    evalin('base',['semilogy(',x,',',y,')'])
```

10.5 小　结

本章中给读者列举了 4 个较为典型的 GUI 程序并对每一个 GUI 的实现过程作了详细的分析。结合前几个章中对 GUI 设计方法的介绍，读者可以较为深入地掌握 GUI 的使用和实现方法。在下一章将向读者介绍 MATLAB 工具箱自带 GUI 的特点和使用方法，读者可以通过该 GUI 了解一个可以投入实际应用的 GUI 应该具备怎样的功能。

第 11 章　工具箱 GUI 的使用

以上章中向读者介绍了 GUI 的整个设计开发过程，本章则要向读者介绍 MATLAB 自带 GUI 工具的特点使用方法。MATLAB 中有许多方便灵活的 GUI 工具，例如 demo、simulink、ident 等等，此处以系统辨识工具箱的 GUI——ident 为例。本章首先向读者介绍 ident 的界面结构和组成部分，然后介绍 ident 的数据管理、模型估计和检查方法。本章假设读者已经有一定的系统辨识专业基础。

11.1　系统辨识工具箱 GUI

1. GUI 界面

系统辨识工具箱提供一个图形用户接口（GUI），该 GUI 支持该工具箱的大部分函数，能够简便地访问 MATLAB 运行过程中创建的所有变量。在 MATLAB 命令窗口中键入：

ident

即可进入如图 11-1 所示的系统辨识工具箱 GUI——主辨识信息窗口 ident。

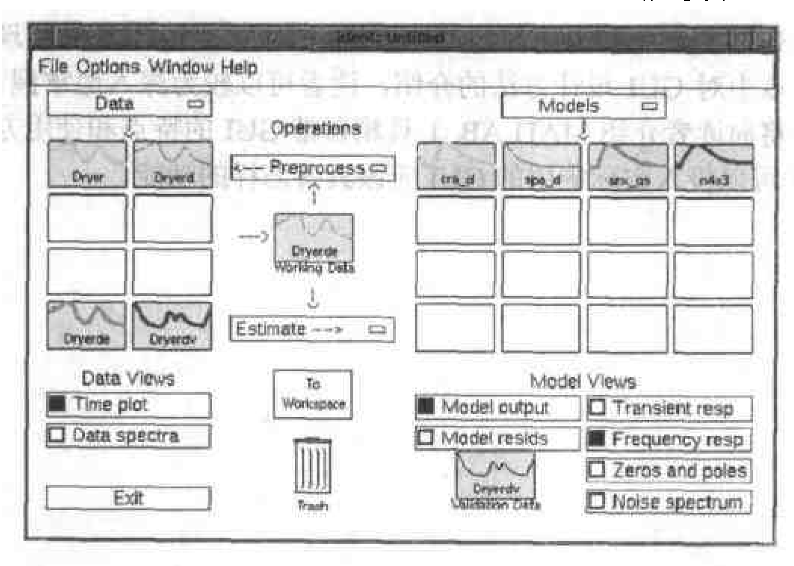


图 11-1　系统辨识工具箱 GUI 外观

下面介绍该窗口的几个组成部分。

2. 模型和数据板

系统辨识过程与数据、模型以及数据建模有关。因此 ident 本质上是由以下两个表支配的：一个包含可获得数据集、以图标表示的数据表；一个包含所创建模型、以图标表示的模型表。这两个表分别称为数据板和模型板。

用户可以通过以下几种方式将数据输入到数据板中：

- 打开以前所保存的运行信息；
- 在 MATLAB 工作平台中导入数据，由弹出式菜单的 Data 选项控制；
- 通过对数据板上的其他数据集进行消除长久趋势、过滤、抽取子集等操作创建新的数据集，具体操作由弹出式菜单的 Preprocess 选项控制。

模型输入模型板的方法如下：

- 打开以前保存的运行结果；
- 在 MATLAB 工作平台中导入新的模型，由弹出式菜单的 Models 选项控制；
- 根据数据进行模型估计，由弹出式菜单的 Estimate 控制。

数据板和模型板可以通过拖放重新布局，也可以同时显示多个板。

3. 工作数据

所有的数据集和模型都是在工作数据集 Working Data Set 中创建的，工作数据集位于 ident 窗口的中心位置，如果要修改工作数据集，从数据板中拖放数据集到工作数据集的图标下即可。

4. 视图控制

在数据和模型板下有一些不同功能的复选框，可以通过使用这些按钮绘制数据集或模型来考察数据集和模型的特征。选择一个数据集或模型的图标，然后点击相应的按钮将会弹出相应的数据图形窗口，从而观察相应的数据特征属性。被选的图标用一条粗线来标示。可以同时观察任意数量的数据或模型对象。

5. 验证数据

对验证数据集使用模型视图复选框 Model Output 和 Model Residuals 可以说明模型的属性。验证数据集就是位于这两个复选框下面的编辑框。同样可以从数据板中拖放数据集到验证数据框中。在辨识过程中通常会使用一个新的数据集来估计一个模型的属性，也就是说使用一个估计模型时未用到的数据集来检验模型的属性。因此，一般验证数据与工作数据有所不同。当然，二者也是兼容的。

6. 工作流

当用户使用导入数据的方法输入数据集时，使用 Data View 来检验数据。用户可能会删除某个数据集中的部分数据或从中选择数据子集，然后用弹出菜单的 Preprocess 属性来进行模型估计和验证，再使用 Estimate 进行模型估计。用户根据自身的外观需要使用 Models Views 来检查获得的模型，任何被检查的模型都将显示其所有属性。这个功能是“现场”级的，所以能够以在线方式对模型和视图进行输入输出检查。用户可以通过分析模型绘图结果重复进行不同模型的构造工作，直至发现满意的模型为止。

7. 外观管理

外观管理包括以下结构内容：

- 管理日志：一般用户很容易忘记曾经执行过的操作。在 ident 中，可以通过双击一个数据集/模型的图标来获得一个完全的日志，该日志给出该对象的创建过程和一些其他的关键信息。通过使用日志还可以修改对象的名称和颜色并添加注释；
- 排列：为了获得一个良好的创建模型和数据集的整体外观，最好能够通过拖放对图

标进行重新排序。在这种方式下，可以执行诸如聚合指定数据集相关模型等操作。用户还可以打开新的数据或模型板进行进一步的图标排序。新板中的图标可以在屏幕中的不同窗口中拖放；

- 运行列表：所有数据集和模型的模型和数据板（包括日志）可以在任意时刻被保存并稍后进行重新装载。这与 MATLAB 驱动命令窗口工作平台的保存/装载方式十分相似。有四个最近运行的运行列表被列在 File 下可以立即打开；
- 清除：数据板和模型板可以放置任意数量的数据集和模型（必要时可以通过板复制获得），同时也可以清除任意一个不再需要的数据集和模型。通过将对象拖到 Trash Can 中来清除对象。双击 Trash Can 图标还可以打开所有可恢复的对象，如果遇到内存不足的问题时可以将这些对象统统删除；
- 窗口控制：对话框和绘图窗口都是由 GUI 的 close 函数妥善管理的，一般 GUI 和窗口管理系统就是管理窗口的最佳方法，所以最好不要将窗口也图标化。

8. 工作变量

正常情况下在 MATLAB 工作平台中是不能获得 GUI 中创建的模型和数据集的。事实上，工作平台根本就不干预 GUI 运行过程中的变量。但是任何时候都可以将对象图标拖放到 To Workspace 中，从而将变量导出到工作平台中，这些变量在工作平台中使用被拖放图标的标注作为变量名。用户可以在工作平台中使用这些变量并将修改后的数据重新导入到 GUI 中。注意模型和数据实际上是作为 idmodel、idfrd 和 iddata 对象导出到工作平台中的。与工作平台中不一样的是，不同的 GUI 数据集和模型对象可以使用同一个名称。

11.2　数据管理

11.2.1　数据描述

在系统辨识工具箱中，信号和观测数据被描述为以下的列向量：

$$u=\begin{bmatrix} u(1) \\ u(2) \\ \cdots \\ \cdots \\ u(N) \end{bmatrix}$$

向量的第 k 个元素 u(k)表示信号在采样时刻 k 的数值。工具箱中通常假设数据是以等距的采样时间进行采样的，采样间隔 T 一般作为函数的参数。通常使用字母 u 表示系统输入，y 表示输出。输入数据有一个矩阵描述，该矩阵的每一列是一个不同通道的输入信号：

u=[u1 u2...um]

多通道的输出矩阵与之类似。在系统辨识工具箱中观测的输入输出数据记录由 iddata 对象描述，该对象是根据数据输出信号由以下语句创建的：

Data = iddata(y,u,Ts)

其中 Ts 是采样时间。iddata 对象还能够在数据插入 GUI 时根据输入输出信号来创建。

11.2.2　输入输出数据插入 GUI

一个数据集需要提供给 GUI 的信息包括：输入输出信号；数据集名称；采样间隔。另外，除了这些基本信息以外，用户还可以提供一些帮助保存记录的属性信息：采样起始时间；输入输出通道名称；输入输出通道单位；输入信号的周期和采样间行为；数据注释。

当用户选择弹出式菜单 Data 的 Import 选项时将会出现如图 11-2 所示的对话框，在这个对话框中用户可以输入 8 个信息条。对话框分为 3 个需要填入的域。如果选择 More 将会出现另外 3 个域。

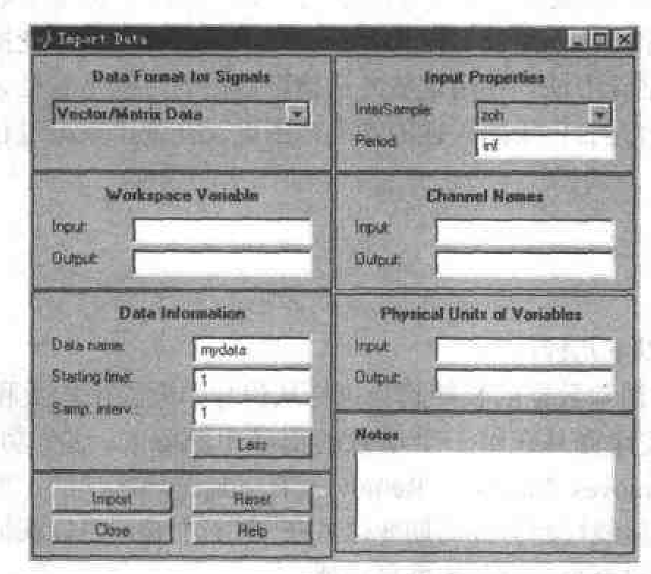


图 11-2　导入数据对话框

- Input 和 Output：可以在此处键入输入和输出变量名。输出输出都应该是 MATLAB 工作平台中的变量。用户可以在这个域中键入 MATLAB 表达式，在数据插入 GUI 之前将会对这些表达式进行求值；
- Data Name：键入在 GUI 中使用的数据集名称。这个名称可以在以后修改；
- Starting time and Sampling interval：填入正确的绘图时间和频率范围。

以下这些内容是可选的：

- Channel names：键入输入输出通道的不同字符串名称，用逗号分开。名称数目必须与通道的数目相等。如果忽略这些内容，GUI 将使用缺省的 y1，y2...和 u1，u2...；
- Channel Units：以相同的格式输入各通道的测量单位。这个单位符合所有根据数据创建的模型，但是只对绘图信息有用；
- Period：如果输入是周期的，在此处键入周期长度。缺省值“Inf”表示非周期输入；
- Intersample：选择输入信号的采样间行为为 ZOH（零阶保持，即输入信号在采样间分段固定）或 FOH（一阶保持，即输入信号在采样间分段线性）或 BL（带宽限制，及连续时间输入信号在 Nyquist 频率上无功）。ZOH 为缺省值。

对话框底部的文本框是注释，用户可以在此处键入任意以保存数据为目的的注释。最后按下 Import 按钮将数据插入 GUI 中。Reset 按钮将清空所有输入。以上描述的步骤将创建一

个 iddata 对象及其所有属性。如果用户已经有一个工作平台中可获得的 iddata 对象，用户可以通过选择对话框顶部弹出菜单的 Iddata Object 选项直接插入该对象。

11.2.3　观察数据

将数据插入 GUI 数据板后的第一件事就是数据检查。使用 Data View 的 Time plot 选项将会显示所选数据集的输入输出信号图形。对于多变量数据可以在绘图窗口的 Channel 菜单项下选择不同的输入输出信号对。使用 zoom 函数（使用鼠标左键绘制矩形）可以更仔细地观察数据的不同成分。如果希望检查数据的频率范围，选择 Data view 的 Data spectra 项。该功能类似于 Time plot，但显示的是信号的范围。缺省情况下将显示数据的周期图，及数据的傅立叶变换的完全平方。使用窗口的 Options 菜单项可以使图形变换为任意所选的频率范围或使用不同的范围估计方法。使用这些方法检查数据的目的是看是否有部分数据不适于辨识、数据信息显示内容是否位于感兴趣的频率范围以及在使用数据进行估计前是否需要进行预处理。

11.2.4　数据预处理

数据预处理方法有以下几种：

- 消除长期趋势：消除数据的长期趋势包括从信号中删除平均数和线性趋势（平均数和线性趋势将被计算并在每一个单独的信号中删除）。这个功能是通过弹出菜单 Preprocess 的 Removes Means 或 Removes Trends 选项实现的。更高级的趋势消除方法（例如删除分段线性趋势或周期变化）在 GUI 中是不可以获得的。通常在进行模型估计前建议用户至少要删除数据平均值；
- 选择数据范围：由于一些不同的意外特性（丢失或错误数据、突发性干扰、改变基准等）导致整个数据集不适于辨识的情况时有发生。此时建议用户将部分测量数据用作估计而另一部分数据作为验证之用。弹出式菜单项 Preprocess 的 Select Range 将打开一个对话框，该对话框能够通过键入数据范围或使用鼠标键框选数据的方法来选择不同的数据范围。对于多变量数据而言，使用部分输入输出信号开始工作是非常有效的。菜单项 Preprocess 的 Select Channels...允许用户选择输入输出信号的子集。由于模型覆盖了不同的数据子集，所以计算数据和模型属性时，输入输出的标号和名称都是不变的；
- 预过滤：通过使用线性滤波器过滤输入输出信号，可以进行诸如在不影响估计模型的条件下删除数据中漂流和高频干扰等操作。这是通过使用主窗口的 Preprocess 的 Filter...弹出菜单项实现的。该对话框与在时域中选择数据范围的对话框非常类似。用户使用一个矩形在范围绘图中标示滤波器的通频带或禁频带，然后将滤波后的数据插入到 GUI 的数据板中。预滤波是删除数据高频噪声的一个好方法，也是一个消除长期趋势（通过切掉通频带的低频）可选的好方法。根据所需的模型使用方法，用户还可以确认模型是否集中在重要的频率范围内。例如，对于一个要被用来进行控制设计的模型，所需闭环带宽附近的频带特性是非常重要的。如果用户希望使用数据同时建立系统的动态模型和干扰模型，建议在模型估计阶段使用滤波器，这是

通过首先选择弹出菜单的 Estimate 的 Parametric Models 选项，然后将估计模型的 Focus 设置为 Filter。预滤波仅仅适用于从输入到输出的动态估计。干扰模型由原始数据决定；

- 重采样：如果数据采样过快，可以进行数据精简，即仅仅选择间隔为 k 的数值。这个功能可以由菜单项 Preprocess 的 Resample 获得。用户还可以通过给定多个重采样因数，使用相同的命令进行插值快速重采样；
- 快速开始：弹出菜单项 Preprocess 的 Quickstart 执行以下一系列的操作：打开 Time plot Data view，通过删除信号平均值生成一个数据集，另外将消除了长期趋势的数据分割为两个数据集，第一个作为工作数据而第二个作为验证数据。生成的三个数据集都被插入 GUI 数据板中。

ident支持包含多个不同实验的数据集。模型的估计和验证都可以使用这样的数据集。这种功能对处理那些描述同一个系统的不同条件下的实验数据非常有用，对处理那些从一个巨大数据集中摘取的数据段也非常有效。多个实验数据能够作为iddata对象插入GUI并在GUI中使用。选择一个多实验数据集的指定段可以通过使用弹出菜单项Preprocess的Select Experiment实现。如果需要在数据板中合并多个数据集，使用Preprocess的Merge Experiment选项。

11.2.5 数据控制步骤

由以上的介绍可以看出，数据控制分为以下几个步骤：

- 将数据插入 GUI 的数据板；
- 绘制数据并仔细检查；
- 通过删除平均值来消除数据的典型趋势；
- 选择用于模型估计和验证的数据段，将这些数据段拖放到 GUI 的相应控件中。

11.2.6 数据仿真

ident 主要针对真实数据集进行操作，本身并不提供仿真综合数据的功能，因而数据仿真要在 MATLAB 命令模式下进行。用户可以选择 Simulink、信号处理工具箱或其他仿真工具箱进行模型仿真，然后将仿真好的数据导入到 ident 中。系统辨识工具箱也包括一些仿真命令。下例说明了 ARMAX 仿真模型的使用方法：

$$y(t)-1.5y(t-1)+0.7y(t-2)=u(t-1)+0.5u(t-2)+e(t)-e(t-1)+0.2e(t-1)$$

以二进制随机信号作为仿真系统输入：

```
model1 = idpoly([1 -1.5 0.7],[0 1 0.5],[1 -1 0.2]);
u = idinput(400,'rbs',[0 0.3]);
e = randn(400,1);
y = sim(model1,[u e]);
```

现在输入 u 和输出 y 可以作为数据导入 GUI，并且可以对其使用不同的模型估计方法。通过将仿真模型 model1 导入 GUI 还可以将其与其他不同的估计模型相比较。

如果要仿真一个连续时间状态空间模型：

$$x=Ax+Bu+Ke$$

```
y=Cx+e
```

使用与上例相同的输入信号，采样间隔为 0.1 秒，在系统辨识工具箱中进行如下操作：

```
A = [-1 1;-0.5 0]; B = [1; 0.5]; C = [1 0]; D = 0; K = [0.5;0.5];
Model2 = idss(A,B,C,D,K,'Ts', 0) % Ts = 0 means continuous time
Data = iddata([],[u e]);
Data.Ts = 0.1
y=sim(Model2,Data);
```

11.3　模型估计与检查

11.3.1　模型估计基础

根据数据估计模型是系统辨识工具箱的主要功能，也是该工具箱能够提供可能性种类最多的操作，因而也是用户最需要执行的操作。所有估计方式都能够通过 ident 窗口的弹出菜单 Estimate 选项来访问。模型总是使用当前工作数据框中的数据集进行估计的。注意区别以下两种不同类型的估计方法：

- 对系统的脉冲或频率响应进行直接估计：这些方法通常被称为非参数估计方式。这些方法除了认为系统是线性结构以外对系统不使用任何结构假设；
- 参数估计方法：这种方法首先假设一个固定的模型结构，然后使用数据对这个结构的参数进行估计。这样的操作根据系统描述的方式不同可以产生很多种可能性。主要的模型结构描述方式是状态空间和多变量微分方程描述。

11.3.2　直接估计方法

1. 脉冲响应直接估计

一个线性系统能够使用脉冲响应 g_k 来描述，这是因为系统具有以下特性：

$$y(t)=\sum_{k=1}^{\infty} g_k u(t-k)$$

这种估计方式是基于这样一个事实的：如果输入 $u(t)$ 是一个脉冲信号，也就是说，当 t=0 时 u(t)=1，而 t>0 时 u(t)=0，那么系统的输出 y(t)将为 y(t)=g。对一个多变量系统来说，脉冲响应 g_k 是一个 ny×nu 矩阵，其中 ny 是输出的数目，而 nu 是输入的数目。该矩阵的第 i-j 个元素就表示第 i 个输出对第 j 个脉冲输入的响应。通过选择菜单项 Estimate 的 Correlation Model，脉冲响应的系数将根据输入输出数据使用称为关联性分析的方法直接进行估计。用户也可以直接在 ident 窗口下键入一个字母 c，该字母是关联性分析方法的热键。

脉冲响应估计的结果放置在模型板中，使用缺省名 imp（可修改）。如果希望检查结果，最好的方法是选择 Model View 的 Transient Response 选项，该选项给出一个估计响应的图形。用户可以在提供的脉冲响应和阶跃响应中选择一种。对于一个多变量系统，微分通道（即从

一个确定输入到一个确定输出的响应）可以由菜单项 Channel 选择；脉冲响应估计的滞后数目（即估计响应的长度）由 Transient Response 的选项之一来控制。

2. 频率响应直接估计

一个线性系统的频率响应是其脉冲响应的傅立叶变换结果。这种系统描述的方法给出大量系统属性的工程内涵。输入输出关系通常写为：

$$y(t)=G(z)u(t)+v(t)$$

其中 G 是传递函数，v 是附加干扰。角频率 w 函数 $G(e^{jwt})$即为频率响应和频率函数。t 是采样间隔。系统的频率响应是使用频谱分析方法直接进行的。首先通过选择菜单项 Estimate 的 Spectral Model，然后选择随之打开的对话框的 Estimate 按钮即可。估计结果放在模型板中，缺省名为 spad。检查结果的最好方法是使用模型视图的 Frequency Response 选项进行模型绘制。这个选项提供许多种控制图形曲线绘制方法的选项。估计响应的频率也可以通过 View 窗口下的 Options 菜单来选择。频谱分析命令同时也估计系统描述中额外干扰 $v(t)$的频谱。估计干扰的频谱由模型视图下的 Noise Spectrum 选项来检查。频谱分析估计结果将作为一个 idfrd 对象来存储。如果用户希望对这些估计进行进一步的操作，可以将模型导出到 MATLAB 工作平台下，然后直接从该对象中恢复响应或使用 Nyquist 或 Bode 命令恢复。可以在对话框中设置两个影响频谱估计的选项：

- 数目 M（延迟窗口的尺寸）：该数目影响估计的频率结果。本质上频率解大约为 2π/M（弧度/采样间隔）。M 的选择同时影响频率解的效果及其变化（波动）情况。一个大的 M 数值将给出一个好的解但是波动大，可信度小。缺省的 M 值一般对系统都比较适合，没有响应尖峰，但对于一些共振系统可能需要一些调节；
- 选择 Black-Tukey 窗口中 spa 方法或光滑的直接傅立叶变换方法 etfe。etfe 对强共振系统非常有效，这种方法在 M 比较大时效率较高。它的缺点在于需要线性的间隔频率值，不能估计干扰的频谱也不提供可信的时间间隔。

  可以在 ident 窗口下使用热键 S 获得当前设置下的频谱分析模型。

11.3.3 参数模型估计

系统辨识工具箱提供大范围的线性系统模型结构，这些结构可以在ident窗口下使用菜单项Estimate的Parametric Models...来访问。该选项将打开一个参数模型对话框，该对话框是一个包括所有待估计参数的基本对话框（如图11-3所示）。

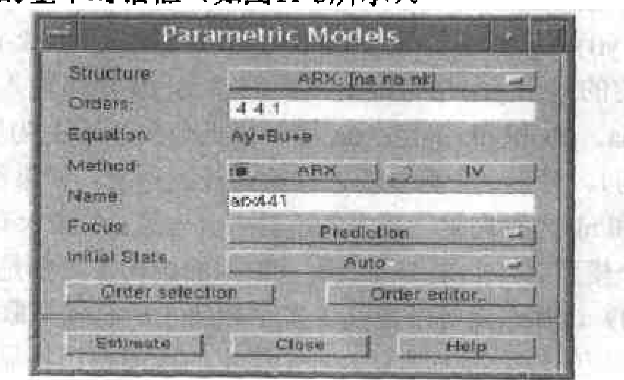


图 11-3 参数模型 Parametric Models 对话框

如果选择Estimate，ident将根据工作数据进行模型的估计。这个模型的结构由弹出式菜单Structure和Orders编辑框共同决定。给出的模型名称由Name编辑框显示。如果用户希望修改模型名称，在按下Estimate按钮前对Name编辑框进行修改。如果希望以后能够导出该模型，那么模型名称中不能有空格。模型结构与Structure所选择的内容有关，相关的阶数说明（通常为整数）将在Order编辑框中显示，通常有6种结构可供选择：ARX模型；ARMAX模型；输出-误差（OE）模型；BJ模型；状态空间模型；用户自定义结构。

用户可以自己填写Order编辑框，也可以选择Order Editor，这将打开另一个与Structure有关对话框，可以在这个对话框中简单地输入所需模型的结束和结构。用户还可以在Order编辑框中输入一个MATLAB工作平台变量名，这个变量取值将与被选结构的基本阶数一致，注意对于状态空间结构和ARX结构，可以输入多个阶数或阶数联合，产生的所有模型都会显示在一个特殊的对话框窗口中，用户可以从中选择一个较为合适的阶数。

常用的参数估计方法是误差预报方法，这种方法简单地选择模型的参数，能够最小化模型预估计输出与测量输出间的差异。所有的模型结构都能够使用这种方法。除了ARX结构，估计过程对其他模型都将会重复地进行数值搜索直至找到最合适的参数为止。为了获得搜索信息并与搜索过程交互，选择Iteration control..按钮。选择一个重复估计过程之后就可以看到这个按钮，该按钮提供控制搜索过程的一些选项。

对于某些模型结构（ARX模型、黑盒状态空间模型），还可以使用基于关联性方法中的仪器变化（IV）方法和子空间（N4SID）方法。通过Parametric Models对话框进行这两个选项的选择。该对话框还有两个弹出式菜单提供一些其他选项：Focus允许用户选择将频率集中在模型预估计或仿真实现的部分；InitialState选择进行初始状态估计或固定初始状态为0。

估计好的模型将被插入ident的模型板中，用户可以检查其属性并使用模型视图复选框选项来进行模型比较。如果仅仅需要观察模型本身，那么双击模型的图标，Data/Model Info窗口随之打开，该窗口将给出用户模型估计的信息。用户可以选择Present按钮在MATLAB命令窗口中列举模型、参数及其标准偏差。如果用户需要对模型进行进一步的操作，将模型图标拖放到To Workspace图标下，然后使用MATLAB及其工具箱命令。

11.3.4　模型结构

1. ARX 模型

这是一种最为常用的模型，是简单的线性微分方程：

$$y(t)+a_1y(t-1)+...+a_{na}y(t-na)=b_1u(t-nk)+...b_{nb}u(t-nk-nb+1)$$

这个方程将当前的输出 y(t)与有限个以前的输出 y(t-k)和输入 u(t-k)联系起来，因而整个结构由三个整数 na、nb 和 nk 决定：na 等于极点个数，nb-1 为零点个数，nk 是系统的纯时间延迟（死区时间）。对于一个典型的采样数据控制系统，如果没有死区，nk 值为 1。对于多输入系统，nb 和 nk 为行向量，第 i 个元素给出与第 i 个输入有关的阶数。

在同时估计多个模型时，如果部分或所有输入的结构参数都是以 MATLAB 分号分隔的向量（例如 na=1:10），那么就等于定义了多个相应于所有模型联合阶数的不同结构。选择 Estimate 后，所有相应的模型都被计算。一个指定的绘图窗口将显示适合于这些模型的验证数据，可以在图中选择任意一个模型插入模型板中。

对于多输入模型，用户可以给每一个指定一个阶数和延迟向量。由阶数和延迟联合生成的模型数量可能会很大，因此可以给所有输入定义同一个向量（例如 nb=1:10），给所有延迟定义同一个向量，这样能够保证只有有限多个对输入具有相同阶数和延迟的模型被计算。

有两种估计 ARX 模型系数 a 和 b 的方法：

- 最小二乘法：求取 a，b 使表达式右边数值减去左边数值结果的平方和最小的方法。这种方法可以通过将 Method 设置为 ARX 来实现；
- 仪器变量法：确定 a，b 使表达式两边的误差与输入的线性组合无关。这种方法通过将 Method 设置为 IV 来实现。

对于一个具有 ny 个输出和 nu 个输入的多输出 ARX 结构，以上的微分方程仍然有效，惟一的变化在于系数 a 变为一个 ny×ny 的方阵，b 变为一个 ny×nu 的矩阵。模型阶数矩阵变为[NA NB NK]，其含义如下：NA：一个 ny×ny 矩阵，第 i-j 个元素表示第 j 个输出与第 i 个输出关联多项式的阶数；NB：一个 ny×nu 矩阵，第 i-j 个元素表示第 j 个输入与第 i 个输出关联多项式的阶数；NK：一个 ny×nu 矩阵，第 i-j 个元素表示第 j 个输入与第 i 个输出间的延迟。阶数编辑器 Order Editor 对话框允许以下的选择：

NA = na-ones(ny,ny)

NB = nb-ones(ny,nu)

NK = nk-ones(ny,nu)

式中的 na、nb 和 nk 是通过弹出菜单选择的。在 MATLAB 命令窗口中构造一个矩阵[NA NB NK]并在 Parametric Models 窗口的 Order 编辑框中输入该矩阵的名称。注意多输出的 ARX 模型不能同时进行多个模型的估计。

2. ARMAX，OE 和 BJ 模型

系统辨识工具箱中有几种模型是从基本的 ARX 模型推出、同时建立干扰微分方程构成的，典型的类型包括 ARMAX，输出-误差 OE 和 Box-Jenkins（BJ）模型。

一个单输出系统的一般输入输出线性模型可以写为如下形式：

$$A(q)y(t)=\sum_{i=1}^{nu}[B_i(q)/F_i]u_i(t-nk_i)+[C(q)/D(q)]e(t)$$

这个结构通过给出时间延迟 nk 和多项式 A、B_i、C、D 和 F_i 的阶数（即从输入到输出的多项式的零极点个数以及从误差到输出的干扰模型的零极点个数）来定义。

很多情况下定义选择的结构有如下的特殊形式：

ARX：$A(q)y(t)=B(q)u(t-nk)+e(t)$

ARMAX：$A(q)y(t)=B(q)u(t-nk)+C(q)e(t)$

OE：$y(t)=[B(q)/F(q)]u(t-nk)+e(t)$

BJ：$y(t)=[B(q)/F(q)]u(t-nk)+[C(q)/D(q)]e(t)$

以上这些变换操作多项式仅仅是一种严格的系统微分方程描述方法，通常并不使用这些形式。举例来说，ARMAX 模型经常写为如下的形式：

$y(t)+a_1y(t-1)+...+a_{na}y(t-na)=b_1u(t-nk)+...+$

$b_{nb}u(t-nk-nb+1)+e(t)+c_1e(t-1)+...+C_{nc}e(t-nc)$

注意 A(q)要符合多项式模型和干扰模型的共同极点（这在干扰对于系统输入来说是封

闭的情况下非常有用)。同样 $F_i(q)$)决定输入多项式的独有极点，D(q)决定干扰的独有极点。介绍这些不同模型的目的在于提供灵活的干扰描述形式，同时允许模型对不同的输入可以有相同或不同的极点。

使用参数模型对话框的 Structure 弹出菜单选择 ARX、ARMAX、OE 和 BJ 结构的一种作为输入模型结构。注意，如果工作数据集有多个输出时，只有第一个选项是可用的：对于时间序列（无输入信号的数据），这些选择中只有 ARX 和 ARMAX 是可用的。多项式的阶数通过 Order Editor 对话框窗口的弹出式菜单来选择。也可以直接在参数模型窗口中的 Orders 编辑框中输入阶数。

多项式系数的估计使用预估计误差/最大可能性的方法，尽量减小以上表达式中误差 e 的大小。有几个控制最小化过程的选项可以通过在参数模型窗口中选择 Options 激活 Iteration Control 来访问。注意这些模型结构只有在标量输入的情况下才能使用。对于多输出模型，使用状态空间结构是一种非常灵活的方式。还要注意不能同时对多个不同的输入输出模型进行估计。

3. 状态空间模型

基本的状态空间模型格式如下：

x(t+1)=Ax(t)+Bu(t)+Ke(t)

y(t)=Cx(t)+Du(t)+e(t)

系统辨识工具箱支持两种参数化的状态空间模型：黑盒自由参数类型和应用程序参数定制类型。

首先讨论黑盒类型。此时最重要的结构指标是模型的阶数，即状态向量 x 的维数。使用 Order Dditor 的弹出菜单选择模型阶数或直接在参数窗口的 Orders 编辑框中输入模型阶数。使用 Order Dditor 其他弹出菜单还可以对所选的模型结构产生影响：固定 K 为 0 将给出一个输出-误差模型，也就是说可以使模型的仿真输出与测量输出之间的差别最小。这种方法建立在干扰为白噪声的假设前提下。

可以给每一个输入选择一个独立的输入延迟。输入延迟是一个 nu 维的行向量 nk。当延迟大于等于 1 时，离散时间模型中的 D 矩阵固定为 0。对于由分段固定输入驱动的物理系统来说，如果没有纯时间延迟，那么将会很自然地假设 nk=1，这也是 nk 的缺省值。

通过使用 MATLAB 分号输入模型阶数向量（例如 1:10）可以同时估计多个模型，此时所有指定的阶数将按照一种系统预先指定的方法进行计算。用户可以通过点击指定的包含模型信息的图形将模型的不同阶数输入到模型板中。

状态空间模型有两种基本的估计方法：

- PEM：这是一种标准的预估计误差/最大可能性方法。这种方法建立在一种标准的反复最小化基础上。反复过程从 n4sid 计算所得的参数值开始。矩阵 A、B、C、D 和 K 的参数化方法是自由选择的。搜索最小值的方法由一些选项来控制，通过 Iteration Control 窗口的 Option 按钮来访问这些选项；
- N4SID：这是一种基于子空间的方法，不需要进行重复搜索。估计结果的质量可能与 N4Weight 和 N4Horizon 选项有关。这两个选项可以在 Order Editor 窗口中访问。如果 N4Horizon 为多行数值，那么模型每一行的相应水平都会使用工作数据分别进行检查，从中选择预估计性能（当 K=0 时取仿真性能）最好的模型，同时显示一个

说明所取模型与模型行函数关系的图形。如果 N4Horizon 编辑框为空，ident 将选择一个缺省值。

4. 用户自定义模型结构

系统辨识工具箱支持用户自定义的任意结构的线性状态空间模型idss。使用模型种类idss，A，B，C，D，K和X0矩阵中已知和未知的参数都能够简单地定义为离散或连续模型中的一种。idgrey对象允许用户使用一个由M文件定义的完全任意的灰盒结构，可以对模型对象属性进行简单地操作。

如果希望在GUI中联合使用上述这些结构，仅仅需要在MATLAB命令窗口下定义合适的结构，然后使用Structure弹出菜单来选择By Initial Model，并在参数模型窗口的编辑框Initial Model中输入结构的变量名，最后选择Estimate按钮。

可以使用系统辨识工具箱模型对象定义以下任意一种模型结构：

- idpoly：为单输出模型创建输入输出结构；
- idss：使用任意的自由参数创建线性状态空间模型；
- idgrey：创建完全任意的线性系统参数化结构；
- idarx：创建多变量 ARX 结构。

另外，所有估计命令都将根据模型估计结果创建相应的模型结构。在参数模型窗口中的Orders（或Initial model）中输入任意模型结构的名称，然后选择Estimate，则模型的结构参数将按照所选的工作数据集进行调整。使用的估计方法是一种标准的预估计误差/最大化可能性方法，该方法将按照某种标准反复地搜索最小值。控制搜索过程的选项通过Iteration Control窗口的Option按钮来访问。注意初始模型的名称必须是一个工作平台或模型板中的变量名。

11.3.5 检查模型

对模型进行估计实际上只是全部工作的第一步，下一步则要进行模型检查、比较并使用新的数据集进行测试。这些工作主要使用6个位于主ident窗口底部的模型视图复选框实现：频率响应、瞬态响应、零极点、噪声频谱、模型输出、剩余模型，通过选择这些功能可以获得各种所需的模型响应或分析图形。另外，用户可以双击模型图标获得模型的文本信息。最后，用户可以将模型导出到MATLAB工作平台中，使用其他命令对模型进行进一步的分析和使用。注意通过频谱分析得到的模型只能够使用频率响应和噪声频谱描述，使用关联性分析获得的模型只能用瞬态响应描述。

这6个视图功能将给出类似的图形窗口，这些窗口有一个包含一些基本功能的公共菜单栏。首先注意这些窗口有一个缩放功能：使用鼠标左键绘制一个矩形，释放鼠标键后该矩形中的内容就会放大。双击后原始的坐标轴刻度将会被保存。对于使用两个坐标轴的图形而言，x轴刻度是指定的。如果不希望使用缩放功能，可以使用Style下的菜单项Zoom使缩放功能无效。其次，如果选择图形中的任意曲线并点击鼠标右键，将使用模型的名称和描述坐标对曲线进行辨识。

双击一个模型图标后，一个数据模型信息对话框将随之出现。该对话框中包含模型的一些基本信息，并给出模型创建日志和与原始相关估计数据集有关的注解。选择该对话框用户

可以实现以下工作：

- 显示相关信息：选择 Present 按钮将在 MATLAB 命令窗口中显示模型的细节。模型的参数及其估计标准偏差和一些其他注解将被显示；
- 修改文本：用户可以简单地在可编辑文本域的任意所需位置处输入任意文本。用户还可以通过编辑文本域来修改模型的名称，模型的颜色也可以在所有绘图中进行编辑，只需在相应的编辑框中输入 RGB 值或颜色名（例如‘y’）。

如果用户有控制系统工具箱，那么还可以在主窗口中看到To LTI Viewer图标。通过将模型拖放到这个图标中可以打开LTI观测器。这个观测器可以控制任意数量的模型，但是要求所有模型有相同的输入输出数量。

下面对这6个视图功能进行一一介绍。

1. 频率响应和噪声频谱

所有估计的线性模型可以写成以下形式：

$$y(t)=G(z)u(t)+v(t)$$

式中G(z)表示系统的（离散）传递函数，v(t)表示额外干扰。系统的频率响应或频率函数是一个角频率 ω 的复数数值函数$G(e^{jwt})$，这个函数经常作为一个波特图来绘制，也就是说分别绘制$G(e^{jwt})$幅值（绝对值）对数和$G(e^{jwt})$相位相对于角频率w对数的两幅图形。估计的干扰频谱v将按照频谱的强度进行绘制。如果数据是一个时间序列y（无输入），那么使用噪声频谱功能将会绘制y而不给出频率响应。

2. 瞬态响应

观察一个模型的动态性能简单而有效的方法就是通过观察其阶跃响应或脉冲响应，即输入为阶跃信号或脉冲信号时模型的输出。通过比较参数估计模型和关联性分析估计模型的瞬态效应可以获得大量的信息。如果二者性能很好地吻合，那么用户可以确信数据的基本特征已经被正确地提炼出来了。许多模型提供一个额外干扰v(t)的描述：

$$v(t)=H(z)e(t)$$

式中H(z)是一个传递函数，描述干扰v(t)是怎样由白噪声e(t)经过系统产生的。如果希望显示H(z)，用户可以将Channel菜单中的输入设置为噪声成分。

3. 零极点

系统的极点是传递函数G(z)分母的根，零点是分子的根。极点对于系统的动态性能有直接的影响。选择零极点视图功能时使用间隔的可信度是非常有用的，这些间隔将清楚地说明哪个零极点将被删除（可信区域重叠），这就意味着系统将使用一个低阶的动态模型。对于多变量系统，每一个输入/输出通道的零极点都将被显示。为了获得所谓的传播零点，用户必须将模型导出并使用系统辨识工具箱提供的tzero命令。

4. 比较标准和模型输出

考察一个模型质量的好坏的一个比较好的办法就是使用新的数据集作为模型的输入，然后将输出的仿真结果与标准输出相比较。这样做将清楚地说明哪些系统特征已经被提炼出来而哪些还没有。使用验证数据框中的验证数据集来进行比较工作。比较结果的不适合度将被显示出来，这是通过计算模型重新生成输出的变化百分率得到的。如果一个模型不符合度为0%，那么仿真输出将给出与标准输出相同的平均方差。如果模型不稳定，或者含有积分器，

又或者时间常数很大，那么即使对于一个非常好的模型而言（至少对于控制目的而言很好），仿真水平和标准输出可能会产生漂移。此时计算模型的预估计输出比仿真输出效果要好。

使用一个k预估计水平（prediction horizon）时的k阶超前预估计输出可由以下方法获得：预估计值y(t)将根据所有可获得的输入u(t)(s≤t)和t-k时间段内的输出y(s)(s≤t-k)计算得到。在不使用任何以前输出信息的仿真情况下，形式上符合k=∞。为了检查模型是否具有用户感兴趣的动态属性，可以将预估计时间水平（kT，T为采样间隔）设置为大于最重要时间常数的数值。这里要注意，不同的模型结构使用过去输出数据信息的方式也不同，具体与干扰模型有关。例如，OE模型（通过固定状态空间模型的k为0并设置输入输出模型na=nc=nd=0得到）根本不使用以前的输出信息，因而仿真输出和预估计输出是一致的。

5. 剩余估计

对于模型：y(t)=G(z)u(t)+H(z)e(t)

噪声源e(t))描述了模型不能够再生的输出部分，称该部分为模型的“左超越”或“剩余”。对于一个好的模型，剩余部分应该与输入无关，否则输出中会含有其他来自于输入而模型没有指出的部分。为了测试这种无关性，计算输入与剩余的互相关函数。此时显示该函数的可信区域也是一个好的选择。对于一个理想模型来说，互相关函数将完全平铺在正延迟的可信范围内。如果在某个延迟k处有一个外部尖峰，那么这就意味着输出y(t)中含有源于u(t−k)的部分而该模型并没有描述出来。测试是使用验证数据进行的，如果不使用验证数据对估计模型进行验证，那么测试效果将非常粗糙。测试也可以使用Model Residuals通过显示剩余的自关联函数（不包括0延迟，因为该函数在此处定义为1）实现。对于一个理想模型，相关函数将位于整个可信区域内。

11.3.6　在 MATLAB 工作平台中的进一步分析

通过在ident主窗口中将其图标拖放到To Workspace下，所有的模型和数据对象都可以导出到MATLAB工作平台中。一旦用户将模型导出到工作平台下，用户就可以使用一些其他命令对模型进行变换、检查并转换为其他格式供其他工具箱使用。以下列出了几个常用的函数：

d2c：转换为连续时间；

ss、idss和ssdata：转换为状态空间描述；

tf、tfdata：转换为传递函数形式；

zpk，zpkdata：转换为零极点形式。

注意命令ss、tf和zkp将模型转换为控制系统工具箱的LTI模型。如果用户有该工具箱，那么该工具箱的LTI命令也可以直接应用于辨识工具箱的模型对象。在MATLAB工作平台下，系统辨识工具箱的所有计算、仿真提取、频率函数、零极点等命令都可以使用。

11.4　实 例 讲 解

【实例】：怎样使用系统辨识 GUI 建立数据较为合理精确的模型？

分析：使用该例的目的是要读者熟悉系统辨识 ident 的整个辨识过程。为此，以 ident 的

一个范例数据集 Dryer 为例，说明从导入数据集开始至模型检查的所有过程及其结果。

解决方法：

步骤一：打开 GUI 界面。在 MATLAB 命令窗口下键入：

```
ident
```

步骤二：导入数据集。在 ident 窗口数据板上方的列表框中选择 Examples，出现如图 11-4 所示的数据导入对话框。

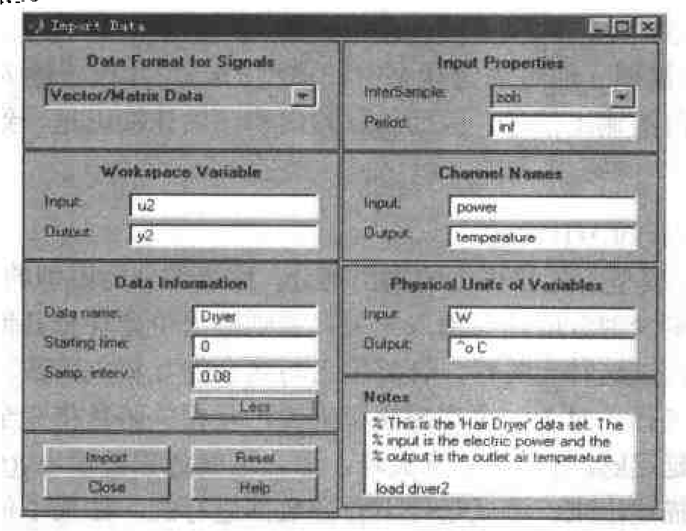


图 11-4　导入 Dryer 数据集时的导入数据对话框

选择 Import 按钮。

步骤三：通过绘图对数据进行检查。选择 ident 窗口中的 Time plot 复选框绘制时域图形（如图 11-5 所示），选择 Data Spectra 绘制频域图（如图 11-6 所示）。

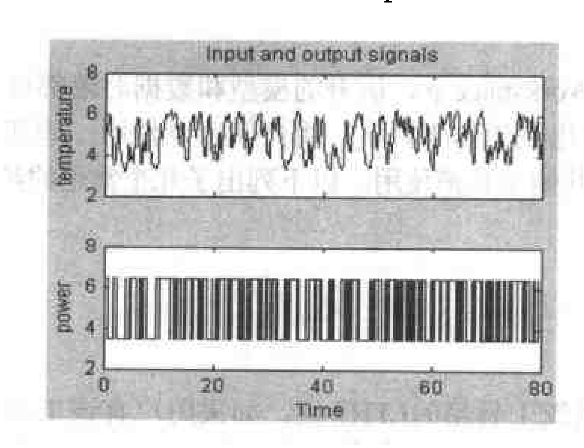


图 11-5　Dryer 时域图形

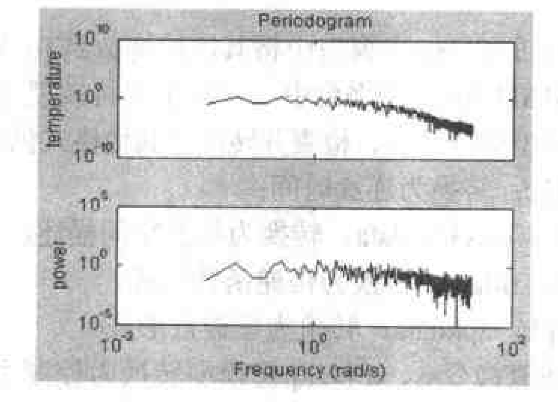


图 11-6　Dryer 频域图形

步骤四：数据预处理。此处采取消除均值预处理法。选择 Operations 列表框的 Removes Means，时域图形变为如图 11-7 所示。

步骤五：选择估计数据段和验证数据段。首先选中数据板中滤波后的数据集（粗线表示选中），然后选择 Operations 下的 Select Range，通过拖动鼠标选择某段数据，点击 Insert，将生成的数据集图标拖到工作数据图标中。重复此操作设置验证数据段。

步骤六：模型估计与检查。在工作数据集下的列表框中选择一种估计方法，例如模型估计方法 Parametric Models，在弹出对话框中将 Focus 设置为 Stability，然后点击 Estimate 按

钮。选择模型的视图复选框 Model Output，绘图结果如图 11-8 所示。从该图形中可以看出，这种估计方法的符合度为 87.1068%。

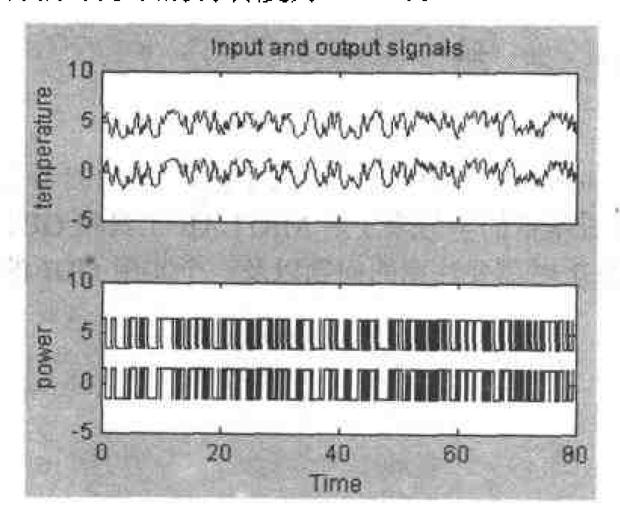


图 11-7 消除均值预处理后的 Dryer 时域图形

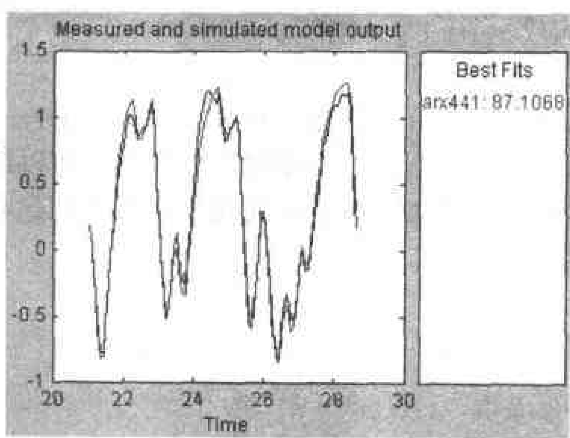


图 11-8 Dryer 模型估计与检查结果

步骤七：导出模型进行进一步分析。将模型板中的 arx441 拖放到 To Workspace 下，然后回到 MATLAB 命令窗口下，输入以下语句将模型转换为零极点形式：

zpk(arx441)

则 MATLAB 的输出结果为：

```
Zero/pole/gain from input "power" to output "temperature":
      0.00084812 (z+0.928) (z^2 + 5.08z + 68.04)
      ---------------------------------------------
      (z+0.3362) (z-0.5804) (z-0.7112) (z-0.1859)
Zero/pole/gain from input "v@temperature" to output "temperature":
0.041717 (z-2.004e-005) (z+2.008e-005) (z^2 - 3.372e-008z + 4.024e-010)
--------------------------------------------------------------------------
             (z+0.3362) (z-0.5804) (z-0.7112) (z-0.1859)
I/O groups:
     Group name     I/O     Channel(s)
     Measured        I          1
     Noise           I          2
Sampling time: 0.08
```

还可以使用以下语句将模型转换为连续时间模型：

d2c(arx441)

输出结果为：

```
Continuous-time IDPOLY model: A(s)y(t) = B(s)u(t) + C(s)e(t)
   A(s) = s^4 + 45.72 s^3 + 698.8 s^2 + 4173 s + 8299
   B(s) = 4.185 s^3 - 1.82 s^2 - 217.9 s + 7632
   C(s) = s^4 + 59.29 s^3 + 1358 s^2 + 1.466e004 s + 6.297e004
   Estimated using ARX from data set Dryerde
```

Loss function 0.00165573 and FPE 0.00176658

11.5 小　　结

本章主要介绍了系统辨识工具箱 GUI——ident 的使用方法，对于控制专业的读者会有较大帮助。对于其他专业的读者来说，可以通过本章的学习来了解 MATLAB 工具箱 GUI 的设计风格和基本思想，从而进一步掌握创建 GUI 时需要注意的问题以及一个实用 GUI 的功能要求。

第 12 章 C/C++图形库使用方法

本章主要面向致力于开发 C 或 C++图形应用程序的读者。本章主要向读者介绍 MATLAB C/C++图形库的组成及组态方法，在此基础上介绍了图形库的使用方法，最后对一些常见的问题做出了说明。

12.1 C/C++图形库介绍

12.1.1 MATLAB C/C++图形库组件

MATLAB C/C++图形库是一个 MATLAB 图形子程序集，通过使用该图形库，用户可以在自己的单机 C/C++应用程序中获得 MATLAB 的绘图和可视化功能。一个单机 C 或 C++应用程序是指一个能够在 MATLAB 编译器环境中独立运行的可执行程序。使用单机应用程序可以方便地对 MATLAB 应用程序进行打包或发布。利用 MATLAB 编译器可以使用 MATLAB C/C++图形库和 MATLAB C/C++数学库，因此用户可以编辑包含曲线、文本、网格和多边形以及图形用户接口组件（例如菜单、按钮和对话框等）的 M 文件。

MATLAB C/C++图形库包括超过 100 个子程序，这些子程序包括：

- MATLAB 6.0 内置的图形函数，例如 surf、plot、get、set 等；
- MATLAB 6.0 常用的图形函数 M 文件，例如 newplot、gcf、gca、gco、gcbf 等。

根据安装系统的不同，MATLAB C/C++图形库所安装的文件也不同。表 12-1 和表 12-2 分别给出了图形库在 PC 和 UNIX 下安装的不同文件类型。表中的<MATLAB>均指 MATLAB 的安装目录。

表 12-1 MATLAB C/C++图形库在 PC 上安装的文件

文 件	安 装 位 置	用 途 描 述
sgl.dll hg_sgl.dll uiw_sgl.dll hardcopy_sgl.dll gui_sgl.dll mpath.dll	<MATLAB>\bin	包含 MATLAB 内核函数和图形函数 M 文件单机版本的共享函数库，所有的 DLL 都是 WIN32 格式的
sgl sglcpp	<MATLAB>\toolbox\compiler\bundles	MATLAB 编译器捆绑文件，包含编译器建立单机图形应用程序所需的所有选项
libsgl.h libmwsglm.h libmwsglm.mlib sgl.def	<MATLAB>\extern\include	图形库头文件和模块定义文件
FigureMenuBar.fig FigureToolBar.fig	<MATLAB>\extern\include	在单机应用程序中 MATLAB 图形窗口使用的预备菜单栏和工具栏文件
flames.m flames.mat	<MATLAB>\extern\examples\sgl	图形库例程的 M 文件和 MAT 文件

表 12-2　MATLAB C/C++图形库在 UNIX 上安装的文件

文　件	安装位置	用途描述
libmwsgl.ext 或 libmwsgl.sl	<MATLAB>/extern/lib/<ARCH>， <ARCH>确定系统体系结构	图形库的二进制文件。这些库对所有操作平台共享。
sgl sglcpp	<MATLAB>/bin/toolbox/compiler/bundles	MATLAB 编译器捆绑文件，包含编译器建立单机图形应用程序所需的所有选项
libsgl.h libmwsglm.h libmwsglm.mlib	<MATLAB>/extern /include	包含内核函数和图形函数 M 文件的图形库头文件
FigureMenuBar.fig FigureToolBar.fig	<MATLAB>/extern /include	在单机应用程序中 MATLAB 图形窗口使用的预备菜单栏和工具栏文件
flames.m flames.mat	<MATLAB>/extern /examples/sgl	图形库例程的 M 文件和 MAT 文件

这里要注意的是，MATLAB C/C++图形库仅包括 MATLAB 图形函数 M 文件的一个子集，并不是 MATLAB 的所有图形函数都可以在 C/C++程序中使用。同时，虽然 MATLAB C/C++图形库支持 MATLAB 6.0 的大多数特征，包括多维数组，单元数组和结构体，但是有一些特征是图形库不能够支持的，这些特征包括：MATLAB 对象、MATLAB 的 java 对象、plotedit 命令、propedit 命令以及某些回调函数。另外，MATLAB 编译器对图形库的支持也是有限的，尤其对于函数 eval 和 input，有一些功能编译器是不能够完成的。所有的限制都可以通过查询 MATLAB 编译器的有关信息来获得。

12.1.2　MATLAB C/C++系统需求

MATLAB 提供了许多共享的动态链接库（dll）来支持 MATLAB C/C++图形库的使用。为了使用 MATLAB C/C++图形库来创建单机的 C/C++应用程序，用户必须使用 MATLAB C/C++数学库 2.1 版本，MATLAB 编译器 2.1 版本和 MATLAB 6.0。另外，用户机器上还必须有一个 ANSI C 编译器。注意，操作系统 IBM RS/6000 不支持 MATLAB C/C++图形库。

12.1.3　MATLAB C/C++图形库组态

在安装完成 MATLAB C/C++图形库之后，用户要使用 mbuild - setup 命令对它进行组态。mbuild 将指定用来编译 MATLAB 编译器生成代码的 C 编译器以及用户应用程序希望链接的库，可以选择仅链接 MATLAB C/C++图形库，也可以选择同时链接图形库和数学库。如果需要在 PC 上的 Windows 操作系统中组态图形库，用户可以在 MATLAB 提示符或 DOS 提示符后输入命令 mbuild 来进行。用户在对图形库进行组态时可以决定 mbuild 使用怎样的选项文件来创建一个单机应用程序。当用户运行 mbuild 时，用户将指定所需的编译器的名称和版本。mbuild 将选项文件定位于指定的编译器并在用户系统目录下创建该文件的一个副本。无论何时通过 mbuild 调用用户的 C 或 C++编译器，MATLAB 编译器都将调用这个文件副本。

下例给出了在 Windows NT 下定义 VC 6.0 作为编译器的整个组态过程，其中下划线部

分为用户的回答。

Please choose your compiler for building standalone MATLAB applications:

Would you like mbuild to locate installed compilers [y]/n?n

Select a compiler:

[1] Borland C++Builder version 5.0

[2] Borland C++Builder version 4.0

[3] Borland C++Builder version 3.0

[4] Borland C/C++ version 5.02

[5] Borland C/C++ version 5.0

[6] Borland C/C++ (free command line tools) version 5.5

[7] Lcc C version 2.4

[8] Microsoft Visual C/C++ version 6.0

[9] Microsoft Visual C/C++ version 5.0

[10] Microsoft Visual C/C++ version 4.2

[0] None

Compiler: 8

Your machine has a Microsoft Visual C/C++ compiler located at D:\Applications\DevStudio.

Do you want to use this compiler? [y]/n y

Would you like to link against the C/C++ Graphics Library? [y]/n y

Please verify your choices:

Compiler: Microsoft Visual C/C++ 6.0

Location: D:\Program Files\DevStudio6

Linking against the C/C++ Graphics Library

Are these correct?([y]/n): y

在上例中，组态结束后 mbuild 将在 c:\winnt\profile 目录下创建一个选项文件。

如果希望在 Linux 下进行组态，则整个过程如下。

mbuild -setup

Using the 'mbuild -setup' command selects an options file that is placed in ~/MATLAB and used by default for 'mbuild'. An options file in the current working directory or specified on the command line overrides the default options file in ~/MATLAB.

Options files control which compiler to use, the compiler and link command options, and the run-time libraries to link against.

To override the default options file, use the 'mbuild -f' command(see 'mbuild -help' for more information).

The options files available for mbuild are:

1: /MATLAB/bin/mbuildopts.sh :

Build and link with MATLAB C/C++ Math Library

2: /MATLAB/bin/mbuildsglopts.sh :

Build and link with MATLAB C/C++ Math and Graphics Libraries

Enter the number of the options file to use as your default optionsfile: 2

12.2　创建单机 MATLAB C/C++应用程序

12.2.1　概述

用户使用 MATLAB 编译器（mcc）来创建一个单机 C 或 C++图形应用程序，在这个过程中，mcc 将执行以下工作：

- 将指定的 M 文件翻译为相应的 C 或 C++源代码模块；
- 产生单机应用程序需要的称之为 wrapper 文件的 C 或 C++其他源代码模块；
- 调用 C 或 C++编译链接器对源代码模块进行编译链接，使之成为一个单机程序。

图 12-1 给出了创建 MATLAB C/C++图形程序的全过程。

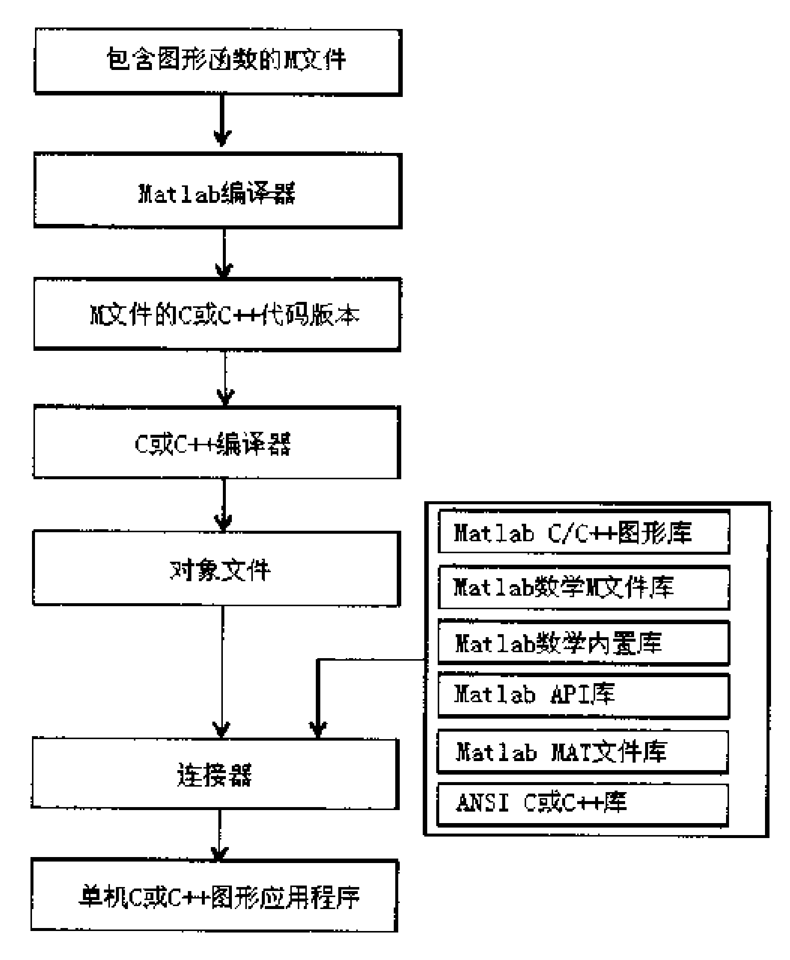


图 12-1　MATLAB C/C++图形程序创建过程

12.2.2　创建单机图形应用程序方法

学习如何建立一个单机图形应用程序最好的方法就是通过一个例子。以下将对 MATLAB 提供的一个演示程序 lorenz.m 进行详细介绍来说明建立图形程序的整个过程。之所以选择

这个例子是因为在该程序中不但用到了一些图形函数，还用到了一些图形用户界面对象。

1. 指定捆绑文件

为了创建一个图形应用程序，用户必须使用并指定 MATLAB 编辑器图形库的捆绑文件。捆绑文件是 ASCII 文本格式文件，该文件包括编辑器的命令行选项和参数。MATLAB 编辑器在创建 C 图形程序和 C++图形程序将支持不同的捆绑文件：如果要建立 C 应用程序则支持 sgl 文件，否则支持 sglcpp 文件。例如，如果需要将 lorenz 应用程序转换为一个单机 C 应用程序，那么在 MATLAB 提示符后键入以下命令：

```
mbuild -setup
    !copy <MATLAB>\toolbox\MATLAB\demos\lorenz.m .
    mcc -B sgl lorenz.m
    !lorenz
```

要注意以下几点：

- 首先要使用 mbuile –setup 建立起创建 C 应用程序的环境。这一步仅在第一次创建 C 图形程序时需要；
- 由于用户可能没有权利在 MATLAB 演示目录下创建一个新的文件，因此有必要将用户程序拷贝到 MATLAB 的当前目录中。上例中使用 DOS 系统的 copy 命令来完成 lorenz 应用程序的拷贝工作；
- 使用-B 标志来调用 MATLAB 编译器，从而指定用于创建 C 图形程序的捆绑文件 sgl。

2. 编译结果

经过编译，MATLAB 编译器将在用户当前的工作目录下产生多个 C 或 C++源代码模块，这些称为包装文件的模块包含一个 C 或 C++程序所必需的成分，例如一个 main 主函数。另外，在用户第一次运行创建一个 C 图形程序时，MATLAB 编译器将在用户的当前工作目录下创建一个名为\bin 的子目录并将在 C 程序运行时需要的 MATLAB 菜单和工具栏图形文件放在该子目录下。在此之后，无论何时调用 MATLAB 编译器，它都会检查这些文件是否存在。

3. 运行 C 图形程序

MATLAB 编译器在用户当前的工作路径下创建 C 图形程序，使之成为一个可执行程序，该程序名与 M 文件名相同，扩展名为 exe。如果用户在 MATLAB 下输入该文件名并在文件名前加一个感叹号“!”，用户就可以运行这个程序。当然，用户也可以在其他环境中运行该程序，然而，一定要确认用户应用程序中使用到的链接共享库位于该环境的搜索路径中，在 Windows NT 中可以通过控制面板的系统属性进行搜索路径的设置。

4. 在 MATLAB 以外的环境中运行 MATLAB 编译器

用户可以在系统提示下在 MATLAB 环境以外调用 MATLAB 编辑器。但是这样做必须要在 MATLAB 命令行中使用-I 选项来指定用户程序所依赖的 M 文件的位置。解决这个问题的一个简单方法是通过运行 mccsavepath 命令为 MATLAB 编译器提供一个路径信息。该命令将在用户当前路径中创建一个名为 mccpath 的路径信息文件。当用户在 MATLAB 环境以外运行 MATLAB 编译器时，编译器自动从该路径信息文件中搜索用户的位置。

12.2.3　改变运行时的行为和外观

一般图形程序的 C 版本都具有与其 M 文件 版本相同的外观，但是，如果在 MATLAB 以外的环境中运行 C 图形程序，不同的 C 程序将会出现一些不同之处。

1. 图形窗口菜单栏选项的变化

单机图形应用程序使用一个特殊版本的、仅包括 File 菜单选项的图形窗口菜单栏。图形库不包括其他标准菜单栏项目，例如 Edit、Tools 和 Help。这是因为单机图形应用程序不能通过这些菜单获得相应的选项支持。图形库还将单机程序不支持的选项从 File 菜单中删除，例如 Page Setup 选项。另外，虽然 Print 选项是支持的，但选择该选项并不显示打印对话框。通过比较图 12-2 所示的 Lorenz 应用程序作为 MATLAB M 文件生成的窗口与图 12-3 所示的 Lorenz 应用程序作为单机应用程序生成窗口的不同可以充分说明这一点。

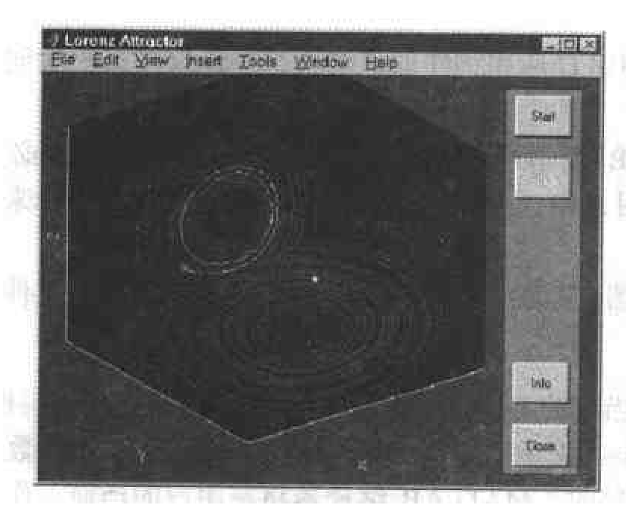


图 12-2　Lorenz 应用程序作为 MATLABM 文件生成的窗口

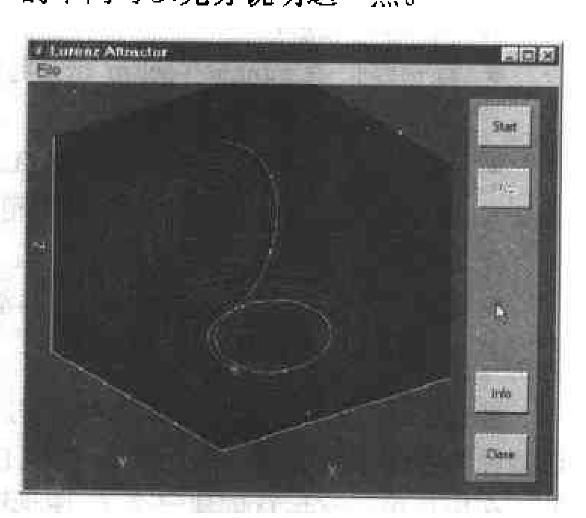


图 12-3　Lorenz 应用程序作为单机应用程序生成窗口

2. 在单机程序中访问帮助

某些 M 文件中使用支持 MATLAB 帮助访问的 GUI 组件。例如，lorenz 应用程序包含一个 Info 按钮来在另外一个窗口中显示 lorenz 函数的 M 文件帮助。单机程序 lorenz 没有访问 MATLAB 帮助文件的能力，如果用户按下 Info 按钮，系统将会报错。

3. Ctrl+C 控制

用户在 MATLAB 中运行图形程序时可以使用 Ctrl+C 来终止无限的循环（例如动画），但是用户运行 C 或 C++单机程序时不支持该操作。

12.2.4　发布单机图形程序

用户可以自由地发布使用 MATLAB C/C++图形库开发的程序。然而，当用户为程序打包时，记住要将以下文件和可执行应用程序包在一起：

- 与用户可执行文件在同一个路径下的\bin 子路径及其所有内容（如果存在的话）；

- 用户程序使用的所有 MEX 文件；
- 所有 MATLAB 数学和图形运行库：MATLAB C/C++图形库已经为用户将所有所需的数学和图形运行库预先打包成一个称为 MATLAB 数学和图形运行库安装的自解压文件，用户只要将这个文件与可执行文件打包在一起即可。在 PC 系统中，这个文件位于 MATLAB 安装路径的\extern\lib\win32\mglinstaller.exe 子路径中。为了使用用户开发的程序，用户在拷贝完成后必须运行 MATLAB 数学和图形运行库安装程序对共享库进行解压并将其安装到用户指定的目录中。另外，用户还必须在使用路径下添加一个 bin/$ARCH 子路径。

12.3 疑 难 解 答

12.3.1 MATLAB 不支持的特征

MATLAB 编译器支持大多数 MATLAB 6.0 的语言特征，例如多维数组，单元数组和结构体等。然而，编译器不支持以下特征：

- 某些特定的 eval 和 input 命令操作；
- MATLAB 对象；
- MATLAB java 对象。

如果用户使用了这些特征，那么系统的错误信息中将会列出用户所使用的错误特征。解决这个问题的惟一办法就是将不支持的特征删除。

12.3.2 编译脚本应用程序产生的错误

如果系统报以下的错误信息：

??? Error: File *"filename"* is a Script M-file and canNot be compiled with the current Compiler.

这就表明用户正在试图编译一个脚本程序。MATLAB 编译器不能够编译脚本应用程序，这是因为脚本是与 MATLAB 基本工作平台相接的，而单机应用程序并不需要也不能够直接访问 MATLAB 基本工作平台。

如果一定要编译一个脚本程序，那么必须将该程序改写为一个 MATLAB 函数，在文件头部定义函数原型。要求用户找出脚本程序中使用到的工作平台变量并将他们定义为全局变量。例如，以下脚本程序中，由函数 figure 返回值定义的变量 f 是保存在基本工作平台中的，而且该变量又要作为一个参数传递给回调函数属性字符串的一部分——my_func 函数。这些都是 MATLAB 编译器不支持的操作。

```
f = figure;
p_btn = uicontrol(gcf, 'style', 'pushbutton',...
      'Position',[10 10 133 25 ],...
      'String', 'Press Here',...
      'CallBack','my_func(f);');
```

下面是根据以上脚本程序改写的函数：

```
function was_a_script()
   global f;
   f = figure;
   p_btn = uicontrol(gcf,'style', 'pushbutton',...
         'Position',[10 10 133 25 ],...
      'String', 'Press Here',...
      'CallBack','my_callback');
```

注意到该函数的以下特点：

- 在文件顶端定义由脚本程序改写的函数原型；
- 该程序将以前在基本工作平台中引用的变量 f 定义为全局变量，这使得回调函数能够对它进行访问。

该程序用一个新的名为 my_callback 的函数代替了回调函数属性字符串中的参考量 my_func，这个新的函数将执行以前回调函数字符串执行的操作。以下是这个新回调函数的代码：

```
function my_callback()
      global f;
my_func(f);
```

12.3.3　处理回调函数问题：函数丢失

创建一个单机程序时，MATLAB 编译器将编译用户在命令行中指定的 M 文件以及用户 M 文件中调用的 M 文件。如果用户程序中出现了在回调字符串中调用函数或将一个字符串作为参数传递给 feval 函数或 ODE 处理器（该字符串在本程序中仅用到这一次）的情况，编译器不会将这个文本字符串视为待编译的函数名，因而不对该函数进行编译，从而导致函数丢失。在这种情况下运行用户程序，交互的用户界面元素（例如按钮）不会响应。关闭用户程序时图形库会报告以下错误信息：

```
An error occurred in the callback : change_colormap.
The error message caught was : Reference to unkNown function change_colormap from
                    FEVAL in stand-alone mode.
```

为了消除这个错误，创建一个包含仅在回调函数字符串中出现的函数列表，并将这个列表传递给%#function，MATLAB 编译器会根据这个列表来搜索相应 M 文件的源代码。例如，下例中在程序 my_test 中调用函数 change_ colormap 将会出现这种问题，为了确保 MATLAB 编译器对文件 change_colormapM 的处理，程序中将函数名使用%#function 列举出来：

```
function my_test()
     %图形库回调函数测试程序
     %#function change_colormap
     peaks;
     p_btn = uicontrol(gcf,'style', 'pushbutton',...
          'Position',[10 10 133 25 ],...
```

'String', 'Make Black & White',...

'CallBack','change_colormap');

除了使用%#function 列表以外，还可以通过在命令行中列出丢失的函数来解决这个问题。

12.3.4 应用程序中无 File 菜单问题

如果用户成功地创建单机应用程序但是 File 菜单没有出现在菜单栏中，这可能表明应用程序\bin 子路径下的菜单栏文件 FigureMenuBar.fig 和工具栏文件 FigureToolBar.fig 不正确。在 MATLAB 安装目录的\extern\include 子目录中找到这两个文件，使用用户程序目录下的相应文件将这两个文件覆盖即可。另外，将用户程序目录下的\bin 子目录删除，然后重新运行 MATLAB 编译器也可以解决这个问题。

12.3.5 依赖于 start-up 文件设置的图形产生的问题

startup.m 文件是 MATLAB 系统的启动文件。如果用户程序依赖于在 startup 文件中设置的图形而用户程序目录下没有该文件，那么用户程序将无法运行。将该文件拷贝到用户的程序目录下即可解决这个问题。

12.3.6 执行图形程序时的问题

如果用户正在使用 VC 5.0 或 MFC 动态链接库，那么在启动一个图形库应用程序时可能会遇到这样一个问题：图形窗口不显示。如果用户是在 DOS 命令窗口下运行该程序的，那么将看到以下的错误信息：

The UIW_SGL.DLL file is linked to missing export MFC42.DLL:####.

或者：

The ordinal #### could Not be located in the dynamic-link libraryMFC42.dll.

如果出现这个问题，用户可以在 windows 系统目录下找到 mfc42.dll 和 msvcrt.dll 文件，将这两个文件复制到 MATLAB 安装目录的\bin\win32 子目录下。同样，如果该应用程序已经发布了，那么该单机程序的使用者也会遇到这种问题，解决方法是相同的。另外，dforrt.dll 或 dformd.dll 文件也会出现同样的问题，解决方法也是相同的。

12.4 实 例 讲 解

【实例】：怎样使用 MATLAB C/C++图形库来编写一个单机 C/C++应用程序？

分析：本例的主要目的是介绍使用 MATLAB C/C++图形库来编写一个单机 C/C++应用程序的完整过程：首先要根据需要编写 MATLAB 图形 M 文件，然后对 C/C++编译器进行组态，使用 C/C++编译器编译 M 文件，最后运行生成的 C/C++可执行文件。本文给出一个

显示动画的图形程序 flames.m 文件，动画数据由 flames.mat 文件提供。该例程位于 extern\example\ sgl 子目录下。

解决方法：

步骤一：编写 M 文件代码。本例使用 MATLAB 例程 flames.m 文件。

步骤二：对 C/C++编译器进行组态。

如果是第一次进行组态，在命令行中输入：mbuild；如果需要再次进行组态，键入：mbulid –setup。在系统提示符后输入 n，系统将显示所支持的 C/C++编译器。在 Compiler:提示符后键入一个数字来选择指定的 C/C++编译器，回车后系统将自动搜索指定编译器所在的位置。确认组态信息即可完成组态工作。

步骤三：编译 flames.m 文件。在命令行中输入：

```
mcc -B sgl flames.m
```

步骤四：执行编译后得到的可执行文件 flames.exe。如果需要在 MATLAB 环境下运行，在命令行中输入：

```
!flames
```

执行结果的一帧如图 12-4 所示。



图 12-4　flames 作为单机程序运行结果

12.5　小　　结

本章介绍了使用 MATLAB C/C++图形库开发 C/C++可执行图形程序的方法。通过使用 MATLAB C/C++图形库，无需再将 MATLAB 开发的 M 文件代码人工地逐条翻译为 C 或 C++ 文件，只要通过简单的编译过程就可以作为 C/C++可执行文件来执行，这将大大方便那些习惯使用 C/C++语言的读者使用现有的 MATLAB 图形程序。

